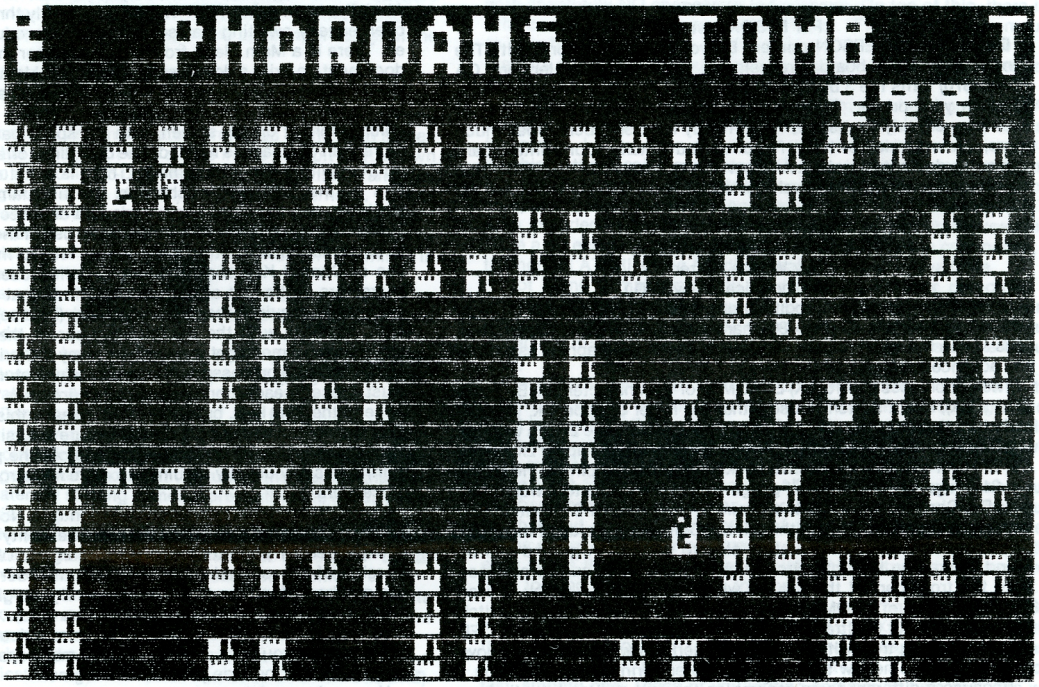
ATARI COMPUTER ENTHUSIASTS
3662 Vine Maple Dr. Eugene OR 97405

MAY, 1984
Mike Dunn & Jim Bumpas, Editors



Sydney Brown: PHAROAH'S TOMB

News and Reviews

by Mike Dunn, co-Editor

For those of you who have been disappointed by the lack of many new programs on our bulletin board, relief is near. Our two 8" drives should be on line by the time you receive the newsletter, adding 1.2 Mbytes of storage — enough for a long time, I hope. Speaking about the bulletin board, when we started it over 2 years ago, we obtained the first **Percom** double-density double drives made. They worked day and night, logging in and uploading programs for several thousand calls a month with no difficulty for all that time. During some routine maintainance work a few weeks ago, they were shut off, and when turned on again, did not work. We sent them back to Percom for repair — they put in all new boards (since ours were "Antiques" which were no longer being made, and they probably forgot how they worked) cleaned and checked the drives, and charged us $59 for two drives!! There are many very fancy drives now on the market, but can they beat this?

We have a number of new disks in the program library, as well as new documentation and a new library list. These include **Education Disks #6 and #7** at $8 each, **Best of ACE #8** at $15, **Action! Disk #1** at $15, and a new **Utility Disk #3** including POLYCOPY which allows you to select specific programs to copy, QuickDOS, various test programs, MemMap, etc. Please send 50c ($1 Overseas) to the Ness' at 374 Blackfoot, Eugene, OR 97404 for an 8-page list just updated. Remember, most of our programs now are documented and are also available on cassette for the same price.

Elsewhere in this issue (I hope!) is a review by Kirt Stockwell on the great O.S.S. Program **BASIC XL** ($99). This very advanced BASIC has many neat fetures, but also is much faster than Atari BASIC. As an example, KONG from the March issue is much faster, smoother and more challenging —without any changes to the programs. You just load it in like Atari BASIC, as it is completely compatible. A cheap way to get almost machine language speed and action without the sweat.

For EPSON owners

Do you own an EPSON printer? Do you want to customize it? The **FingerPrint** add-on board (Dresselhaus, 837 E. Alosta, Glendora, CA 91740 ($70) turns the 3 buttons (Online, FF, LF) into a powerful way to program your printer. Change to compressed, emphasized, italic or even the very tiny "subscript" print handy for disk labels. Does much more, and very nice to have. Requires you to take your printer apart, and some dexterity required. Another way to customize your EPSON is described in the CSRA Newsletter as quoted from the Charlotte PC Newsletter:

EPSON MX-80 Modificaton

by Wayne Setzer

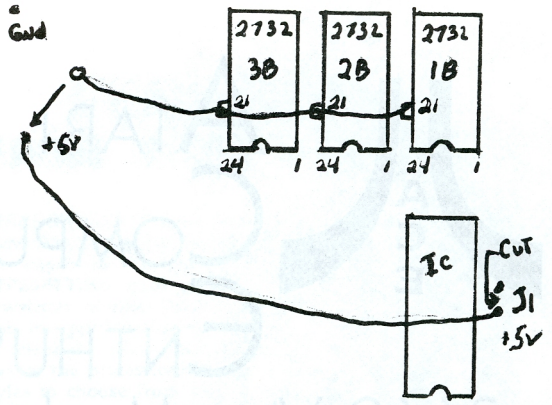
(Article modificated by M.D.)

This article describes modifications to the Epson MX-80 to allow you to use both the Graftrax-80 character set as well as the Graftrax Plus set on the same printer, so you can use block graphics as well as the advanced features of the Plus.

The Epson uses 3 2716 (2K) EPROM's for the character set. By using 2732's (4k), and a switch to use either the lower or upper 2k, you can switch to the character set you want. First program the 2732's with the Graftrax-80 in the lower 2k and the Graftrax-Plus in the upper 2k. Install the EPROM's in their respective sockets, leaving out pin 21. Tie all the pin 21's together with a small gauge wire (a wire wrap tool works fine —M.D.). Connect pin 21 to the common post of a SPDT switch, and mount switch on the back of the printer. Connect one side of the SPDT switch to ½5 volts and the other side to ground. (A ground can be found marked on the right circuit board, and 5 Volts can be obtained from the front part of the cut J1, which must be cut if you use an EPROM instead of a ROM-M.D.). To use the Graftrax-80, set the switch to ground, and for Graftrax-Plus to + volts. Be sure to turn printer OFF before changing the switch setting. (FingerPrint works with the above modification, somehow sensing the different printer codes —M.D.).

Because a different team assembles the text part and the listings part of the newsletter, two articles last month did not have the listings printed due to a lack of communication. They are in this issue— **Device Handlers by Greg Menke** and the **TidBits** article by Dale Lutz. Sorry!

Also, the Global Change program last month had an omission: change Line 800 to read at the end "THEN 920".



BUMPAS REVIEWS

Eastern House Software (3239 Linda Dr., Winston-Salem, NC 27106) has done Atari, Inc. a BIG favor. They have removed one of the most important reasons one might want to keep an 800 rather than buying an XL.

MONKEY WRENCH II is now available in cartridge form for Atari XL computers. The price will be $50 as of June 1, 1984 (the announced date of availability). We'll be sure to let you know how well it works as soon as we've seen a copy.

LETTER WIZARD (Datasoft, $50) is the latest word processor for the Atari. I've been using **TEXT WIZARD** for nearly three years now, so I've been eagerly awaiting Datasoft's improved product. And it is improved. There is a Main Menu from which the user may proceed to Edit a document, to Print a document, or to go to a Menu of disk file management functions including: Formatting a disk, renaming, deleting, locking and unlocking files. The disk directories are displayed in double-column format on the screen, with the space bar used to continue if more than one page is needed. The disk directories may also be called up from within the editing mode. There is a status window at the bottom of the Edit screen showing the file name, amount of RAM available, and the number of words and disk sectors in the file.

Many program features are applied by use of the three function keys in combination with other keys. Letter Wizard moves away from the use of Control Key combinations to permit users to embed control characters in the text of documents. The format for these embedded commands requires the Atari Logo key (inverse) to type the letters "ch", together with the decimal number of the control code. This feature allows me to get more use from my IDS printer than is possible with most word processors. A very powerful feature. The manual says I must precede each control code with (inverse) "ch27" (the decimal value of the ESC key), but for the IDS at least, this code is not helpful. Only the control code of the printer feature I'm using is required. Since the inverse key must be toggled on and off, the program features requiring this key usually require two more key-strokes than other word processors.

Letter Wizard adds two new cursor controls: Now the user can jump to the beginning or end of any line. And flexibility in handling margins is expanded by permitting "relative" margins. Without resetting the margins of the printed page, you now may set off the margins of paragraphs within your text. The temporary margins may be inside or outside the margins of the document as a whole. A very sensible and easy way to handle this problem. No more calculating how wide the paper is. Now you need only decide how many spaces the margins will move, inside or outside your other margins.

You no longer have to worry about having the cursor at the top of the file for saving and loading. If you want to save less than all of a document into a file, separate commands are available to save everything below the cursor. In the same manner, files may be appended below the cursor with another command. If you make a mistake when you delete a block of text, a new command permits you to restore the deleted block. This is a handy feature!

Now for the bad news. The program does not function with any double density disk drive. I can't help think this is a severe marketing error. Double density drives are flooding the market for the Atari. This program requires users to take a step backwards. And after I've converted all my text files to double density, too! Less important, Letter Wizard has abandoned the support for double-column printing. My guess is the greater printer support achieved in Letter Wizard has made it necessary to drop this feature.

Printer support includes three models of Epson, Atari printers, Centronics, Microline, NEC, C.Itoh, and Gemini. The "other" category is the one I must use for the IDS. I still don't seem to be able to underline or right-justify proportional print. With Letter Wizard, I gain expanded print.

There's even a bit of chrome you might appreciate: Screen color and intensity can be selected by the user. It appears to permit one to cycle through all 256 colors. You can make it a green screen, or an amber screen, or a screen of your own choosing. Fun to play with, you'll probably tire of it since the default screen comes up the same each time the program boots. Also on the disk is a BASIC program to use to convert FILEMANAGER 800+ (Synapse Software) files for direct use within Letter Wizard files.

Letter Wizard is an excellent, full-function wordprocessors for any purpose, and is an exceptionally good value for the price.

MASK OF THE SUN (Broderbund, $40) is a 4-disk graphic adventure with a very imaginative storyline. You have a lot of open territory to explore in your jeep, tooling around in Central Mexico. You have at least 3 Aztec ruins to explore, each of which has more than one level. There are secret doors, one-way doors, objects and artifacts to collect.

The goal of the game is to find the Mask of the Sun and decipher the cure which will save your life. Page flipping of graphics screens is used to simulate animation. The program replies sarcastically to some of your moves. You have an enemy trying to stop you, or beat you to your goal. And there are dangers aplenty to spoil your progress. This is an excellent game, and a fantastic value for the price. Adventure gamers will be lost for hours playing this one.

DIMENSION X

DIMENSION X(Synapse Software $30), aside from being a graphics spectacular is also very playable and "wears well." It is sort of a cross between **STARRAIDERS** and Synapse's own game, **ENCOUNTER**.

The objective of **DIMENSION X** is to eliminate enemy saucers before they eliminate you and/or get control of the "Capital" (represented as a square on a grid map in the upper right corner of the screen display).

As the game begins an intialization screen appears which give you choices as to the level you wish to play,the strength of your shields, and the number of alien saucers you wish to encounter. When you make these choices you are immediately projected to the main screen display — and what a display it is! You are in the cockpit of your own ship looking over a scrolling 3-D landscape with mountains and tunnels in the background and a blue sky and clouds overhead. Moving your joy stick forward moves your ship forward over the checkerboard surface of the planet. You may also move back or from side to side by moving the joy stick. This visually gives you the effect of "zooming" in and out as well as traveling at great speed.

Enemy saucers appear from time to time and fire at your ship. You either dodge their shots while firing back or your shields are destroyed and the game ends.

The top of the screen has, in addition to (1) the grid map, (2) a scanner to help you locate enemy saucers, (3) a fuel gauge, (4) a shield status indicator, as well as (5) a message board which warns you of various hazards. Despite all of these "aids" and the graphic display, the screen is not cluttered.

To move from one grid to another it is necessary to pass through tunnels at the edge of each sector. In the tunnel the display changes. What you see is laterally shifting planes representing the edges of the tunnels and a series of gates which you either go over or under. If you hit the sides of the tunnel or do not avoid the gates you will sustain damage. At the novice level these gates are not difficult — but at higher levels with increased speed I got frustrated at times.

One of the squares on the grid map has an "F" on it. This represents where you need to go to automatically replenish your fuel supply and to repair all damages sustained in battle.

If you succeed in destroying all of the saucers you will be given a rating and classification on a final screen (shades of **STARRAIDERS**).

I succeeded once in reaching a class 1 status mostly by luck. The highest level, labeled "Expert", is too fast for me to master as yet.

This is a good game. It has had good staying power and variety. There are a lot of subtleties to **DIMENSION X** which are discovered only by playing it. I do very well on the battle field but not so hot maneuvering in the tunnels. The graphics alone make this game a worthy addition to the Atari repertoire.

— Graham Smith

BASIC XL

I don't impress easily. A product has to be quite good for me to give it the green light. In the case of **BASIC XL** ($99 OSS), I am impressed all over myself. I received my computer programming training on Mainframe computers. Most of these have extremely good editors and very powerful versions of BASIC. Needless to say, I was rather shook when I bought my ATARI and found many of the high level commands I was used to were missing. And, while the ATARI editor is the best of all home computers, it still lacks many features.

The first thing the programmer will notice is the improved editor. Automatic line numbering, built-in renumber, and block line deleting make it much easier to modify programs. Another powerful aid to programming is the TRACE function. The trace function is not be particularly useful in graphic programming, as it kills all graphic modes and functions only in GR.0. On the other hand, if you are going buggy trying to find just where the extra characters crept into that string, this is the ticket.

There are other nice features to the editor, and some which are useable through the editor OR in deferred mode in programs. First, and probably the most desirable is the DIR(ectory) command. If you are like me you probably don't update your disk labels often enough. This makes it fun trying to find which disk has what. Also accessible are most of the standard DOS commands, which saves PLENTY of time if the program you are developing does any file manipulation, such as creating data files. All in all, the editor is among the nicer I have seen. The only more powerful editors are those on larger machines which COMPILE the basic programs prior to running them.

One thing you will notice about BASIC XL right away is it runs **FAST**. Without any modifications to existing programs, any largeish program will run noticeably quicker. As an example, I booted MASTERTYPE(tm) with BASIC XL in the computer. While the program was thinking it was running at 25 WPM, I counted 34 WPM. Again, this is with absolutely NO changes to the program. On top of this, there is a nifty command which comes in very handy. "FAST" tells the computer to do a basic pre-compile on the program. I won't take the time to explain its operation here, but it can make a very significant differance in speed.

One of the things that bothered me the most when I got my ATARI was the lack of string arrays. Once I got used to building my own string arrays, I felt better and found that if I worked at it I could to anything other Basics could do. BUT, it took a lot of extra programming. BASIC XL has all of the string handling features of the best microcomputer basics. Not just string arrays, but MID$, LEFT$, and RIGHT$. Also included is the FIND command, which makes string searching quite a bit easier. This does not force you to use these commands, as the normal ATARI string handling features are also available.

Not being a mathematician, and avoiding number crunching as scrupulously as possible, I couldn't see any changes in the math functions. So I compared the command lists in the ATARI BASIC and BASIC XL manuals. Sure enough, no changes. (At least none I could find).

For people like myself who just LOATHE mucking about with the details of setting up and manipulating PLAYER MISSILE graphics, there are a whole flock of commands to simplify the process of using PMG. This should open the use of PMG to many programmers who have avoided it in the past.

Let's not forget the error codes. ATARI BASIC simply gives you a number when it encounters an error. BASIC XL not only gives you a number, but a short explanation of the type of error encountered. Those of you who don't goof up enough to have memorized all of the error codes (like I have) will appreciate this feature. It will save you from looking everything up in the manual.

Those of you who intend to do any serious programming in BASIC should strongly consider picking up BASIC XL. Not only do the features make programming easier, but the increased speed in many cases is enough to make the difference between a clunky game and a fast, smooth one. Those of you writing application-type software will find it faster and easier. I have used all of the enhanced Basics that I have seen for the ATARI. Of them all, BASIC XL is the best. In my opinion, this is a MUST for BASIC programmers.

Now, is anybody listening at ATARI. I have talked with Bill Wilkinson at OSS. He assures me BASIC XL could be implemented in the new 1450XLD (if there really is to be such a thing) without making any mods to the operating system OR the circuit boards. If you plan on putting out an advanced personal computer, you should use the most advanced implementation of BASIC available. I seriously believe the search for a better Basic ends here.

— Kirt Stockwell

The Computer Faire

The 9th annual West Coast Computer Faire was quite a bit of fun, as well as highly informative. Everybody from hobbyists to educators, to programmers and equipment designers can find something of interest there.

This year the Faire experienced possibly the most drastic change since its inception. Many people in the computer industry, as well as potential buyers of home computers were not expecting ATARI to live through the past year. I am happy to report that ATARI Home Computer Division seems to be healthy.

The ATARI display at the Faire was first rate. The display cases were really nice looking, and the set-up was more tempting to investigate. The new ATARISOFT group was in attendance, with about 35 percent of the total space ATARI had rented. They were displaying programs on ATARI and COMMODORE computers. The quality of the games was quite good, and the selection promises to grow rapidly. When I first heard about the ATARISOFT group last June at the CES in Chicago, I thought it was a good idea but wondered whether ATARI could pull it off. It seems the new management knows its stuff.

It seems there has been a change in the marketing attitudes of the major computer and softwre vendors. Most of you have probably noticed that in the past year or two there have been far fewer computer stores around, not to mention the dwindling numbers of those who specialize in any one home computer. The trend in marketing is "a computer can be sold the same way as a toaster or microwave oven". Personally I am not convinced this is an intelligent selling program, but it seems to have caught on quite well. With the major mass market retail chains selling computers, the manufacturers feel very little need to spend the bucks necessary to impress the little people at the Faire. I think this is a mistake. What the major companies are doing is trying to sell computers with absolutely no promise of meaningful customer support.

The software companies are another story. Any of you who spend any time looking at software in the stores will have noticed that very few new programs of any type have been released fo any of the home computers lately. Piracy has a lot to do with this. The companies lose so much money on piracy they are afraid to release anything worthwhile. They are also spending inordinate amounts of money trying to protect their sotware from piracy. Personally, I don't think this is the answer. In some cases, the protection is costing the companies more than the author will receive in royalties. The authors also suffer from the delayed release dates, as they don't receive any meaningful compensation until the program is released and begins selling.

I should mention that OSS was quite well represented. I had several opportunities to talk to Bill Wilkinson, who is a very knowledgeable and interesting person. Bill is expecting several developments in the next year. Among these are a parallel drive for the XL series and possible a parallell HARD DRIVE (10 meg or so) for the XL computers. Bill also says BASIC XL is currently outselling ACTION! by almost 2 to 1. I expect to see this change this year. Action! is not only easy to learn, but is incredibly powerful.

Publishers were in evidence all over the show. One area of the home computer field which seems not to have had a bloodbath this year is the magazines. Antic, Analog, Compute and a host of new magazines were all well represented, as well as magazines spcializing in the other home and personal computers. Time prevented me from talking at length with many of these people, but I did get to spend some time with Jim Caparrel of ANTIC. We discussed the attitudes of users and user groups, and the growing sophistication of computer owners in general.

In past years there have been many small companies represented at the Faire producing specialty software or unusual peripherals. This year they all seemed to be missing. One of the reasons for this may be the cost of the facilities. Here in the Pacific Northwet (sic), housing can be built for a cost of about 6 to 9 dollars per square foot. When you pay that cost, you OWN the building. The cost of renting the floorspace at the computer faire has risen to $14.50/square foot. For this you get 5 days' worth of rental, electricity, and rudimentary security. Many small companies cannot foot a bill like that, especially when you consider the cost of staying in San Francisco for 5 days or a week. Many of these small companies also seem to not have lived through the past year.

All in all, the Computer Faire was instructive and enjoyable for me, but then I wasn't looking for anything specific. Don Marr, owner of Royal Software based here in Eugene, was so disappointed he didn't stay through the whole Faire. Most dealers I talked to felt the same. They came looking for new software and/or hardware they could retail. For the most part they went away empty-handed. The few companies who were showing anything worth selling tended to pick up lots of new outlets, as the dealers found little new to pick up.

The overall thrust of the Faire this year seemed to be toward small business computers. There were more IBM PC clones than I ever imagined existed (it's beyond me why anybody wants to copy that obsolete turkey). The PERSONAL COMPUTERS (as opposed to home computers) were well represented, with software and hardware vendors all over the place.

It is obvoius there is begining to be a very noticeable disparity between the Home computers and Personal computers. It is likely there will, witin the next few years, be separate shows for Home and Personal computers.

A big surprise at the Faire was running into Tom Mannus of Neanderthal Software. Tom visited Eugene last summer, and showed a prototype of "Turbo 810", a board to give the Atari 810 drives true double-density capacity. Tom insists the board is still under development and will be released by this summer.

One nice thing about the Faire was the weather behaved beautifully. Coming from a L-O-N-G wet winter in Eugene, Oregon, I was happy as a clam to get a little sun.

— Kirt Stockwell

THE SHAPE OF THINGS

by Ruth Ellsworth

This month's article began with one idea in mind and ended with another. The promised article on more advanced string concepts in ATARI PILOT will appear next month illustrated by several routines which can be added to this program.

PILOT is a very powerful language, more powerful than most realize, and can do most of the things it is possible to do in BASIC through peeking and poking. One of the things it can not do is an array. It also can use only twenty-six numeric variables. If this has been a problem, take heart. There are several ways around these problems, and this program demonstrates one.

We have been delighted with the articles in ANTIC MAGAZINE by Phil and Kathy Bergh dealing with character sets and player missile graphics for PILOT. This program is a utility for creating new characters to be used in text or as "player-missiles" in graphics. In order to make the program assign one (and only one) value to each square of the grid I needed sixtyfour variables. Psuedo-arrays and extra variables can be obtained in PILOT by poking values into unused memory locations. For this particulor program we decided it was easier and less typing to assign each square of our grid a memory address. Since the lower numbered end of page six in memory does not seem to be used in PILOT we chose those numbers to use as our storage locations. One of the reasons we decided to solve our problem this way is that, although the memory locations can be added, subtracted, etc.; we could not get our computer to C(#X = number) + (#Y = number):store value at specified location. The equation should work, and we can only guess it did not because the values were joystick location points.

The use of empty memory locations opens up a world of possibilities in PILOT. For beginners like we are, there are two options which can immediately be put to use. Those locations can be assigned variables, such as #A, to be used in numeric functions, or they can be used to store a boolean 0 or 1, and peeked for use in programs where a true or false value is needed.

The other concept illustrated by this program is the writing of data to disk storage. This month we are demostrating numeric variables only (next month will be string variables). Lines 2040 to 3040 demonstrate the technique used to store numeric variables to disk (cassette can be used if the D: is changed to C: when the program is typed). It is possible to Read and Write to a device in PILOT by using the READ: and WRITE: commands. There are several rules, however. WRITE: and READ: must be separated by a CLOSE: command. The device must be specified followed by a :. The data must have a name. Numeric and string variables cannot be mixed. Numeric variables can only be stored and retrieved separately. (String variables are easier to store, but advanced string techniques which we will show next month are needed to separate them upon retrieval.)

The retrieval routine listing is designed to be added at the beginning of the progam in which the new character is to used. The shortest and easiest demonstration program for character sets I am aware of is printed in the August 1983 ANTIC by Phil and Kathy Bergh. It should be loaded first and renumbered so the retrieval routine will appear the beginning. A line 105 J:*BEGIN and 1 *BEGIN must be added, and the numbers in lines which read C:@B#W = number must be changed to read C:@B#W = variables #A to #H (e.g. C:@B#W = #A).

During the run of this program typing C will save the character, and typing E will erase a square on the grid. Additional characters can be saved by renaming the filename in lines 2040 to 3040. The lines at the beginning of the program will initialize the location of the joystick, and the memory locations so typing run will allow the program to be reused without any glitch.

In the event someone is curious as to why the grid does not appear in the middle of the screen, next month we will use string techniques to place the numbers on the screen near the grid rather than at the bottom of the screen. Until then "get in shape" with the utility. Character sets are a lot of fun.

```
50 PA:100
60 R:THIS PROGRAM MUST BE MODIFIED TO
ALLOW THE USE OF CHARACTER UTILITY
65 R:CHANGE ALL #A BELOW TO #R OR ANY
LETTER VARIABLE NOT BETWEEN A TO H INC
LUSIVE.
66 R:ADD LINES AS EXPLAINED IN MAY ACE
 NEWSLETTER.
70 C:#Z=@176/1024*3
80 C:#Z=#Z*1024
90 C:#W=#Z
100 C:#A=#Z
110 C:#Y=@B756*256
120 C:@B756=#A/256
130 C:#X=0
140 *MOVEIT
150 C:@B#Z=@B#Y
160 C:#Y=#Y+1
170 C:#Z=#Z+1
180 C:#X=#X+1
190 J(#X(1024):*MOVEIT
200 C:#W=#W+(44*8)
210 C:@B#W=7
220 C:#W=#W+1
230 C:@B#W=15
240 C:#W=#W+1
250 C:@B#W=124
260 C:#W=#W+1
270 C:@B#W=12
280 C:#W=#W+1
290 C:@B#W=63
300 C:#W=#W+1
310 C:@B#W=109
320 C:#W=#W+1
330 C:@B#W=57
340 C:#W=#W+1
350 C:@B#W=0
360 T:
390 T:Lori Louise London Library
400 E:
```

Superload by Terry Barker
Atari Computer Club OK City

```
10 REM THESE ROUTINES WRITE A PROGRAM
20 REM WHICH, WHEN ENTERED, WILL INSTA
LL THE SUPERLOAD
30 REM UTILITY AT THE ADDRESS POINTED
TO BY LOCATIONS
40 REM 132 AND 133
50 CHKSUM=0
60 FOR I=0 TO 110:READ X:POKE 1536+I,X
:CHKSUM=CHKSUM+X:NEXT I
70 IF CHKSUM<>14988 THEN PRINT"SORRY,
ONE OF YOUR DATA STATEMENTS IS INCORRE
CT.":END
80 OPEN #1,8,0,"D:SLDATA
90 PRINT#1,"FOR I=0 TO 110:GET#7,X:POK
E 1536+I,X:NEXT I
100 FOR I=0 TO 110:PUT #1,PEEK(1536+I)
:NEXT I
110 PRINT#1,"X=USR(1536+100,100)
120 PRINT#1,"ADML=PEEK(132)+256*PEEK(1
33)
130 PRINT#1,"X=USR(1536,1536,ADML+1,10
0,1)
140 PRINT#1,"CLOSE#7:END
150 CLOSE#1
160 END
170 D.160,0,104,104,133,97,104,133,96,
104,133,217,104,133,216,104,133,219,17
0,104,133,218
180 D.104,104,208,43,162,2,177,96,201,
64,48,3,24,105,9,41,15,202,240,9,10,10
,10,10,133,220,200,144,233
190 D.5,220,170,152,72,74,168,138,145,
216,104,168,200,196,218,208,214,96,224
,0,208,4,196,218,240,247,177
200 D.96,145,216,224,0,208,4,200,24,14
4,236,200,208,233,202,230,97,230,217,2
4,144,225
210 D.104,104,170,104,168,138,162,134,
76,129,168
```

```
5 REM YOU MUST ENTER "D:SLDATA" BEFORE
 RUNNING THIS PROGRAM!
10 DIM A$M$(100)
12 REM -- CLEAR SCREEN --
15 PRINT CHR$(125)
17 REM -- GET ADDRESS OF SUPERLOAD --
20 ADML=PEEK(132)+256*PEEK(133)+1
25 REM -- ADDRESS OF SCREEN --
30 ADSCR=PEEK(88)+256*PEEK(89)
35 REM -- INPUT DATA (IN HEX) --
40 D.73757065726C6F6164
45 REAM A$M$
47 REM -- PACK AND MOVE --
50 X=USR(ADML,ADR(A$M$),ADSCR,LEN(A$M$
),0)
55 REM -- MOVE PACKED DATA --
56 REM -- DOWN SCREEN --
60 FOR I=1 TO 24:X=USR(ADML,ADSCR,ADSC
R+I*41,LEN(A$M$)/2,1):NEXT I
70 POS.2,15
99 END
```

```
5 REM YOU MUST ENTER "D:SLDATA" BEFORE
 RUNNING THIS PROGRAM!
10 DIM A$M$(124)
12 REM -- CLEAR SCREEN --
15 PRINT CHR$(125)
17 REM -- GET ADDRESS OF SUPERLOAD --
20 ADML=PEEK(132)+256*PEEK(133)+1
25 REM -- ADDRESS OF SCREEN --
30 ADSCR=PEEK(88)+256*PEEK(89)
35 REM -- INPUT DATA (IN HEX) --
40 A$M$="A90085D885D968686885D66868C96
0100A186940C960300338E960A2030A26D9CAF
0031890F785D8A5D91865D685D9A007B1
45 A$M$(LEN(A$M$)+1)="D8295591D888F003
1890F460
47 REM -- PACK AND MOVE --
50 X=USR(ADML,ADR(A$M$),ADR(A$M$),LEN(
A$M$),0)
55 REM -- MOVE PACKED DATA --
56 REM -- DOWN SCREEN --
60 Y=PEEK(106):Y=Y-4:POKE 106,Y-1
65 GR.0
70 X=USR(ADML,224*256,Y*256,128*8,1)
75 X=USR(ADR(A$M$),Y,ASC("A"))8*8,1)
76 POKE 756,Y
99 END
```

Dale Lutz

```
0 REM LISTING 1 for Dale Lutz Tidbits4
1 REM SEE ACE NEWSLETTER APRIL ISSUE
 PAGE 4
5 REM OS/A+ 2.10 COMMAND CHANGER
10 DIM A$(3)
20 BASE=5988
30 FOR A=BASE TO BASE+41
35 B=B+1
40 ? CHR$(PEEK(A));
45 IF B>2 THEN B=0:GOSUB 100
50 NEXT A
60 ? "THAT'S IT, NOW WRITE OUT A DOS F
ILE   TO KEEP YOUR CHANGES ":END
100 ? "-->ENTER YOUR ALTERNATE CHOICE"
;:INPUT A$
105 IF LEN(A$)<3 THEN RETURN
110 FOR C=1 TO 3:POKE A-(3-C),ASC(A$(C
,C))
115 NEXT C
120 RETURN
```

Sydney Brown: Pharoah's Tomb

```
0 G=0:U=1:Q=2:SE=U:GOSUB 32000:GOSUB 9
000:DIM CP(8),CT(5),CR(8),D(8)
1 REM ********************************
2 REM **       ACE NEWSLETTER       **
3 REM **    3662 VINE MAPLE DR      **
4 REM **     EUGENE, OR 97405       **
5 REM **    MAY 1984  $12 YEAR      **
6 REM ********************************
7 REM **          TOMB              **
8 REM **           by               **
9 REM **   Sydney Brown             **
10 REM ********************************
190 POKE 708,136:POKE 709,222:POKE 710
,6:POKE 711,24:GOSUB 3000:LEV=1
195 GOSUB 8000:? "{":GOSUB 2000:M=SCR+
42:POKE M,186:POKE M+1,187:CY=G:N=U:NU
=G:KP=LEV*2+2:GOSUB 510:L=18:MAD=U
196 MN=1:FOR W=38 TO 40-KP STEP -1:POS
ITION W,0:? "_";:FOR WW=15 TO G STEP -
0.5:SOUND G,7,10,WW:NEXT WW:NEXT W
200 ST=STICK(G):DI=G:PE=G:IF ST=15 THE
N 220
201 IF ST=14 THEN PE=PEEK(M-39):DI=-40
:IF SCR>LL THEN 204
202 IF ST=13 THEN PE=PEEK(M+41):DI=40:
IF SCR<UL THEN 204
203 GOTO 206
204 IF PE=G THEN SCR=SCR+DI:POKE D2,IN
T(SCR/256):POKE D1,SCR-256*PEEK(D2)
206 IF ST=11 THEN PE=PEEK(M-1):DI=-2
208 IF ST=7 THEN PE=PEEK(M+U+Q):DI=Q
210 IF PE=G OR PE>190 THEN POKE M,G:PO
KE M+U,G:POKE M+U,G:M=M+DI:POKE M,L:PO
KE M+U,L+U
212 IF PE=191 THEN GOSUB 760
214 IF K<LL AND M=LL+196 THEN GOSUB 75
0
216 IF PE>191 THEN 770
220 IF KP>18 THEN 770
230 IF STRIG(G)=G THEN GOSUB 800
250 N=N+U:IF N>5 THEN N=U:POKE 77,G
255 MAD=MAD+U:IF MAD>200 THEN N=MN:IF
MAD>242 THEN MAD=U
281 IF D(N)<>G THEN GOSUB 450
285 IF CY=G AND NU<5 THEN GOSUB 500
290 IF CY>G THEN CY=CY+U:SOUND G,14-CY
,8,10:GOSUB 300
291 IF MAD=200 THEN GOSUB 390
294 IF PEEK(764)=33 THEN ON SE GOSUB 5
50,570
295 IF PEEK(K)<>191 THEN GOSUB 700
296 IF PEEK(53279)=5 THEN GOSUB 850
297 IF PEEK(53279)=6 THEN 790
298 IF L=186 THEN L=184:GOTO 200
299 L=186:GOTO 200
300 IF CY/Q=INT(CY/Q) THEN POKE P,CR(4
):POKE P+U,CR(Q*Q)+U:GOTO 305
301 POKE P,CR(5):POKE P+U,CR(5)+U
305 IF CY<12 THEN RETURN
350 N=NU:CP(N)=P:CY=G
355 R=INT(3*RND(G))+U:SOUND G,14,10,15
:CT(N)=CR(R)
359 POKE P,CT(N):POKE P+U,CT(N)+U:SOUN
D G,G,G,G:GOTO 400
390 MN=INT(5*RND(G))+U:RETURN
400 R=INT(Q*Q*RND(G)+U):ON R GOTO 405,
406,407,408
405 PE=PEEK(CP(N)-U):IF PE=G OR PE>183
 THEN D(N)=-Q:GOTO 410
406 PE=PEEK(CP(N)+U+Q):IF PE=G OR PE>1
83 THEN D(N)=Q:GOTO 410
407 PE=PEEK(CP(N)+40):IF PE=G OR PE>18
3 THEN D(N)=40:GOTO 410
408 PE=PEEK(CP(N)-40):IF PE=G OR PE>18
3 THEN D(N)=-40:GOTO 410
409 GOTO 405
410 IF PE>183 AND PE<188 THEN 770
450 PE=PEEK(CP(N)+D(N)):IF PE>183 AND
PE<188 THEN 770
451 IF PE>G AND PE<223 THEN 480
452 SOUND Q,255-7*N,10,3
455 CP(N)=CP(N)+D(N):POKE CP(N),CT(N):
POKE CP(N)+U,CT(N)+U:POKE CP(N)-D(N),G
:POKE (CP(N)+U)-D(N),G
459 SOUND Q,G,G,G:RETURN
500 CY=U:NU=NU+U:R=INT((5-LEV)*RND(G))
+U:ON R GOTO 501,502,503,504
501 P=LL+228:RETURN
502 P=LL+874:RETURN
503 P=LL+260:RETURN
504 P=LL+602:RETURN
510 K=INT(440*RND(G))*2+LL+U:IF PEEK(K
)>G THEN 510
515 POKE K,191:FOR W=15 TO G STEP -0.2
:SOUND G,9,6,W:NEXT W:POKE 764,255:RET
URN
550 POKE M,G:POKE M+U,G:POKE 712,14:FO
R W=U TO 7:SOUND G,7,10,10:SOUND G,G,G
,G:NEXT W:POKE 712,0
560 M=INT(440*RND(G))*2+LL:IF PEEK(M+1
)>G THEN 560
565 POKE M,L:POKE M+U,L+U:FOR W=123 TO
 7 STEP -Q:SOUND G,W,8,10:NEXT W:SOUND
 G,14,10,15:FOR W=G TO 14:NEXT W
566 SOUND G,G,G,G:KP=KP+U:POKE LL-KP,6
3
568 IF K>LL THEN POKE K,G
569 GOTO 510
570 POKE 712,14:FOR W=U TO NU:POKE CP(
W),G:POKE CP(W)+U,G:FOR WW=15 TO 7 STE
P -U:SOUND G,14,8,WW:NEXT WW:D(W)=G
575 NEXT W:POKE 712,G:NU=G:CY=G:SOUND
G,G,G,G:KP=KP+U:POKE LL-KP,63:IF K>LL
THEN POKE K,G
599 GOTO 510
600 FOR W=255 TO Q STEP -Q:SOUND G,W,1
0,10:SOUND U,W-U,10,10:POKE 711,W:FOR
WW=U TO 7:NEXT WW:NEXT W
605 SOUND G,G,G,G:SOUND U,G,G,G:POKE 7
11,24
610 FOR W=U TO NU:POKE CP(W),G:POKE CP
(W)+U,G:FOR WW=15 TO G STEP -0.5:SOUND
 G,41-WW,8,WW:NEXT WW
620 NEXT W:SOUND G,200,12,15:FOR W=Q*Q
 TO Q STEP -U:POSITION 35,W:? "    ";:
IF W=U+Q THEN POSITION 36,W:? "\]";
630 FOR WW=U TO 123:NEXT WW:NEXT W:FOR
 W=15 TO G STEP -0.2:SOUND G,200,12,W:
NEXT W:FOR W=U TO 5:D(W)=G:NEXT W
690 RESTORE 3900:FOR YY=U TO 12:READ F
,D,T:SOUND G,F,10,15:SOUND U,F+U,10,15
:FOR W=U TO D/Q:NEXT W
691 SOUND U,G,G,G:SOUND G,G,G,G:FOR W=
U TO INT(T/Q):NEXT W:NEXT YY:LEV=LEV+U
:NU=G
692 FOR W=U TO 123:NEXT W:? "{":IF LEV
>Q+Q THEN LEV=Q+Q:W=1
693 IF LEV>U THEN POKE 710,182:POKE 70
8,136:IF LEV>Q THEN POKE 710,134:POKE
708,184:IF LEV>U+Q THEN POKE 710,54:PO
KE 708,184
699 GOTO 799
700 FOR W=7 TO 77 STEP Q:SOUND G,W,10,
10:NEXT W:SOUND G,G,G,G:KP=KP+U:POKE L
L-KP,63
709 GOTO 510
750 SCR=LL:POKE D1,104:POKE D2,INT(SCR
/256)
752 FOR W=14 TO 8 STEP -Q:FOR WW=15 TO
 G STEP -U:SOUND G,W,10,WW:NEXT WW:NEX
T W:POKE LL-KP,G:KP=KP-U
754 IF KP<2 THEN 600
755 GOTO 510
760 FOR W=15 TO G STEP -0.25:SOUND G,7
,10,W:NEXT W:K=LL-40:RETURN
770 FOR W=U TO 21:RR=INT(254*RND(G)):S
OUND G,RR/Q,8,15:POKE 712,RR:FOR WW=U
TO 14:NEXT WW:NEXT W:FOR W=15 TO G STE
P -0.2
772 SOUND G,RR/Q,8,W:NEXT W:POKE 712,0
773 RESTORE 3900:FOR YY=U TO 11:READ F
,D,T:SOUND G,F/2,10,15:FOR W=U TO D:NE
XT W
774 SOUND G,G,G,G:FOR W=U TO T/Q:NEXT
W:NEXT YY:READ D:SOUND G,D/2,10,15:FOR
 W=1 TO 123:NEXT W
775 FOR W=15 TO G STEP -0.2:SOUND G,D/
2,10,W:NEXT W:W=U
780 W=W+U:IF W=70 THEN POSITION 3,0:?
"   THE END   ";
781 IF W=140 THEN POSITION 3,0:? "PRE
```

```
55 START ";
782 IF W=210 THEN POSITION 3,0:? "PHAR
OAH5  TOMB";
783 IF W=280 OR W=300 OR W=320 OR W=34
0 OR W=360 THEN POSITION 3,0:? " C O M
 P C O  ";:IF W=360 THEN W=U
784 IF W=290 OR W=310 OR W=330 OR W=35
0 THEN POSITION 3,0:? "
";
785 IF PEEK(53279)=5 THEN GOSUB 850
786 IF PEEK(53279)<>6 THEN 780
790 ? "K":POKE 710,6:POKE 708,136:LEV=
U
799 SOUND G,G,G,G:SCR=LL:POKE D1,104:P
OKE D2,INT(SCR/256):? "K":GOSUB 8000:G
OTO 195
800 IF STICK(G)=13 AND SCR<UL THEN DI=
40:GOTO 830
810 IF STICK(G)=14 AND SCR>LL THEN DI=
-40:GOTO 830
820 IF STRIG(G)=U THEN GOTO 849
829 GOTO 800
830 SCR=SCR+DI:GOSUB 849:FOR W=1 TO 21
:NEXT W:GOTO 800
849 POKE D2,INT(SCR/256):POKE D1,SCR-2
56*PEEK(D2):RETURN
850 IF SE=U THEN SE=Q:POSITION 19,0:?
"S";:GOTO 860
855 SE=U:POSITION 19,0:? "T";
860 FOR WW=15 TO G STEP -0.5:SOUND G,2
55,10,WW:NEXT WW:RETURN
2000 POSITION 0,1:? "VWVWVWVWVWVWVWVW
WVWVWVWVWVWVWVWVWVWVWVWVWVWVW    VW     V
W      VW           V^^^^WVW         VW
  ";
2001 ? "  VW  VW  VW  VW  V^^^^WVW  VW
VWVWVWVWVW  VW  VW  VW  VW  V^^^^WVW
VW      VW       VW  VW    VW  VW";
2002 ? "VW  VW   VW        VWVWVW  VW
VW       VWVW  VWVW  VWVWVWVWVW  VW  VW
VWVWVWVWVWVWVW         VW          ";
2003 ? "VW              VWVWVWVWVW  VW
  VW  VW  VWVWVWVW  VWVW  VWVW
VW  VW              VW  VW  VWVW  ";
2004 ? "VWVWVWVW  VWVWVWVWVWVW  VWVW
    VWVW        VW         VW        VW
VWVW  VWVW  VW  VW  VW  VW  VWVWVWVW";
2005 ? "  VW  VWVW        VW  VW  VW
       VWVWVW  VW  VWVWVWVW  VW
VWVWVWVW              VW  VWVW    VW  ";
2006 ? "VWVWVW          VWVWVWVWVW  VWVW
VWVW  VW      VW  VWVWVWVW          VW
VW      VWVWVW  VW       VW     VW  VWVW
";
2007 ? "VWVW  VW       VW  VW  VW  VW
        VWVW        VW         VW  VW
  VWVWVWVW  VWVW  VW  VW  VW  VW  VW
";
2008 ? "VW  VW          VWVW       VW
      VW          VW        VW  VWVWVWVWVWVWVWVW
VWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVW";
2009 POSITION 0,0:? "  PHAROAH5  TOMB
 T":IF SE=2 THEN POSITION 19,0:? "S";
2020 RETURN
3000 ? "K":POSITION 7,0:? "ace eugene"
,"presents":FOR W=DL+7 TO DL+18:POKE W
,Q*Q:NEXT W
3001 POSITION 3,2:? "VWV V V  W  VWV V
WV  W  V V V VWV         V V V V VWV V V
 V V VWV V V    V         VWV VWV V V V
WV ";
3002 ? "V V V V VWV  VWV       V       V
V VWV VW  V  V VWV V V     V       V
V V V V V V  VWV V V V V  VWV";
3004 POSITION 7,9:? "VWVWV  VWVWV  V
 V  VWVW                    V     V    V  V
W WV  V  W                          V";
3006 ? "V  V V V  VWVWV
 V    V    V  V    V    V    V
 V       VWVWV  V  V  VWVWV";
3010 FOR XX=U TO Q:RESTORE 3909:FOR YY
=U TO 12:READ F,D,T:SOUND G,F,10,10:SO
UND U,F+U,10,7:FOR W=U TO D/3:NEXT W
3020 SOUND G,G,G,G:SOUND U,G,G,G:FOR W
=U TO T/3:NEXT W:NEXT YY:NEXT XX
3099 FOR W=U TO 200:NEXT W:? "K":FOR W
=DL+7 TO DL+18:POKE W,5:NEXT W:RETURN
3900 DATA 86,49,21,75,49,21,72,140,1,7
2,140,1,86,140,14,86,49,21,75,49,21,72
,49,21,72,49,21,64,49,21
3901 DATA 64,49,21,86,140,21
3909 DATA 86,49,21,75,49,21,72,140,1,7
5,140,1,86,140,14,86,49,21,75,49,21,72
,49,21,64,49,21,75,49,21
3910 DATA 72,49,21,86,140,21
3911 DATA 243,245,247,111,113,190,208,
201
8000 RESTORE 3911:FOR W=U TO 7:READ D:
CR(W)=D:D(W)=G:CP(W)=G:NEXT W
8010 RETURN
9000 DL=PEEK(560)+256*PEEK(561):FOR W=
U TO Q:POKE DL+W,PEEK(DL+W+Q+U):NEXT W
:POKE DL,71:POKE DL+Q+U,6:SC=104
9005 SC5=PEEK(DL+5):POKE DL+Q*Q,69:POK
E DL+6,SC5:POKE DL+5,SC:D1=DL+5:D2=DL+
6:FOR W=19 TO 24
9010 POKE DL+W,PEEK(DL+W+10):NEXT W:FO
R W=7 TO 18:POKE W+DL,5:NEXT W:SCR=SC+
256*SC5:LL=SCR:UL=SCR+400:RETURN
32000 DIM PR$(50):CB=PEEK(106)-4:POKE
106,CB:PR$(U,41)="hh, h, hh,  ,   ,  
   , NHPV MFOhdTPP  ":GRAPHICS 6
32005 POKE 82,G:POKE 752,U:? :? "     
   SYDNEY.H.BROWN       1983          ":R
ESTORE 32760
32010 Z=CB*256:A=USR(ADR(PR$),Z,4):FOR
W=Z+432 TO Z+511:READ D:POKE W,D:NEXT
W:FOR W=888 TO 967:READ D
32020 POKE W+Z,D:NEXT W:POKE 756,CB:RE
TURN
32760 DATA 242,242,243,0,171,255,255,0
,171,255,255,0,242,242,243,0,10,1,85,6
9,69,15,12,60,128,0,84,132,68,240,60,0
32761 DATA 2,0,21,17,17,15,60,0,160,64
,85,97,81,240,48,60,0,0,0,10,42,170,17
0,42,128,160,168,128,160,168,168,160
32762 DATA 63,63,63,63,51,63,63,63,252
,204,252,48,60,48,60,0
32764 DATA 0,3,12,63,255,243,63,15,48,
252,255,255,204,240,252,192,60,255,63,
51,15,15,3,0,240,204,252,252,240,252
32765 DATA 207,252,60,31,175,63,15,3,0
,1,0,0,192,240,255,252,64,16,51,221,63
,59,63,63,63,15,0,192,3,15,63,255,252
32766 DATA 240,3,14,3,15,59,235,239,51
,192,176,192,240,236,235,251,204
```

May Meeting

May 9th 7:30PM
South Eugene High
Cafetaria

The string in Line 32000 of Pharoah's Tomb by Sidney Brown contains the following characters (I use the convention of "C" for Control Key, "I" for inverse):

h, h, I-C-E, I-O, I-C-E, I-N, h, h, I-C-E, I-T, I-J, I-C-., I-C-E, I-L, I-), I-C-I-C-E, I-M, I-", I-A, I-SPACE, I-C-., I-1, I-L, I-C-Q, I-N, I-H, I-P, I-V, I-F, I-M, I-F, I-O, I-h, I-d, I-T, I-P, I-P, C-., I-C-.

-- J.B.

Stan Ockers: Johnny's Paintbox

```
; JOHNNY'S PAINTBOX
; Stan  Ockers  3-84
; Written in Action! (c) 1983 ACS
;
; ACE NEWSLETTER, 3662 VINE MAPLE DR
; EUGENE, OR 97405 MAY 1984 $12 YEAR
;
MODULE
  BYTE j,k,m,paint,consol=53279,key=764,
    under,stk,v,w,byt,sel,x,y,x1,y1,
    x2,y2,r,spd
  BYTE ARRAY hue=[0 2 3 9 10 11 8],
    sign=[18 9 65 8 79 63]
  INT delx,dely

PROC Prtchar(BYTE byt)
  CARD st,pos
  IF byt<32 THEN st=(byt+64)*8+57344
  ELSEIF byt<96 THEN
    st=(byt-32)*8+57344
  ELSE st=byt*8+57344 FI
  FOR k=0 TO 7
  DO
    byt=Peek(st+k) j=7 pos=1
    WHILE pos<256
    DO
      IF byt&pos THEN color=hue(paint)
         ELSE color=0 FI
      Plot(v+j,w+k)
      j==-1 pos==*2
    OD
  OD
RETURN

PROC Ding(BYTE pitch,CARD dly)
  BYTE loud
  CARD wait
  FOR loud=0 TO 15
  DO Sound(0,pitch,10,15-loud)
    FOR wait=1 to dly DO OD OD
  Sndrst()
RETURN

PROC Mode() ; Print mode symbol
 v=70 w=89 byt=sign(sel) Prtchar(byt)
 Ding(sel*10+30,600) key=255
RETURN

PROC Init()
  Graphics(23) Poke(752,1) Poke(623,128)
  Poke(87,10) Poke(704,12) Poke(705,70)
  Poke(706,56) Poke(708,46)
  Poke(709,214) Poke(710,134)
  Poke(712,102) x=40 y=40 delx=0 dely=0
  color=2 Plot(0,0) DrawTo(79,0)
  DrawTo(79,88) DrawTo(0,88)
  DrawTo(0,0)
  FOR m=1 TO 6 DO
  v=8*m+8 w=89
  paint=m Prtchar(48+m) OD
  sel=0 Mode() x=40 y=40 under=0 spd=5
RETURN

PROC Hues() ; Change color using keys
  IF key=31 THEN paint=1 Mode()
  ELSEIF key=30 THEN paint=2 Mode()
  ELSEIF key=26 THEN paint=3 Mode()
  ELSEIF key=24 THEN paint=4 Mode()
  ELSEIF key=29 THEN paint=5 Mode()
  ELSEIF key=27 THEN paint=6 Mode()
  ELSEIF key=50 THEN paint=0 Mode() FI
RETURN

INT FUNC Abs(INT n)
   IF n<0 THEN RETURN( -n ) FI
   RETURN( n )
; From Action! Programmers Disk

PROC Circle(INT r)
 INT Phi,Phiy,Phixy
  Phi=0 x1=r y1=0
  DO Phiy = Phi + y1+y1 + 1
     Phixy = Phiy - x1-x1 + 1
  Plot(x+x1,y+y1) Plot(x-x1,y+y1)
  Plot(x+x1,y-y1) Plot(x-x1,y-y1)
  Plot(x+y1,y+x1) Plot(x-y1,y+x1)
  Plot(x+y1,y-x1) Plot(x-y1,y-x1)
  Phi = Phiy  y1 = y1 +1
    IF Abs(Phixy)+0<Abs(Phiy) THEN
       Phi=Phixy   x1 = x1 - 1  FI
  UNTIL y1 > x1  OD
RETURN

PROC Brush() ; Fill in area with color
  j=x WHILE Locate(j-1,y)=0
  DO j==-1 OD color=0 Plot(x,y)
  color=hue(paint)
  WHILE Locate(j,y)=0 DO
  k=y WHILE Locate(j,k)=0
  DO Plot(j,k) k==+1 OD k=y-1
  WHILE Locate(j,k)=0
  DO Plot(j,k) k==-1 OD
  Plot(j,y) j==+1
  OD under=color
RETURN


PROC Dly(CARD wait)
  DO wait==-1 UNTIL wait=0 OD
RETURN

PROC Draw()
  stk=Stick(0) stk==!15  delx=0 dely=0
  IF stk&1 THEN dely=-1 FI
  IF stk&2 THEN dely=1 FI
  IF stk&4 THEN delx=-1 FI
  IF stk&8 THEN delx=1 FI
  x==+delx y==+dely
  IF x>78 THEN x=1 FI
  IF y>87 THEN y=1 FI
  IF x<1 THEN x=78 FI
  IF y<1 THEN y=87 FI
  IF Delx<>0 OR Dely<>0 THEN
  under=Locate(x,y) FI
  color=9 IF under=9 THEN
  color=2 FI Plot(x,y) Dly(500*spd)
  IF Strig(0)=0 THEN color=hue(paint)
  ELSE color=under FI
  Plot(x,y) Dly(500*spd)
RETURN

PROC Rectangle()
x1=x y1=y DO
color=0 Plot(x,y) DrawTo(x1,y)
DrawTo(x1,y1) DrawTo(x,y1) DrawTo(x,y)
stk=Stick(0) Hues()
IF(stk&1)=0 AND y1>1 THEN y1==-1 FI
IF(stk&2)=0 AND y1<y THEN y1==+1 FI
IF(stk&8)=0 AND x1<78 THEN x1==+1 FI
IF(stk&4)=0 AND x1>x THEN x1==-1 FI
color=hue(paint) Plot(x,y) DrawTo(x1,y)
DrawTo(x1,y1) DrawTo(x,y1) DrawTo(x,y)
IF key=33 THEN
   IF x1>x OR y1<y THEN under=color FI
   EXIT FI OD
RETURN

PROC Letters()
key=255
DO IF key<255 THEN
      x=v y=w Hues() v=x w=y
      IF key=12 THEN EXIT FI
      IF key=52 AND v>8 THEN v=v-8 FI
      byt=Peek(65342+key) key=255
      IF byt<128 AND v<70 AND w<80
      AND (byt>38 OR byt=32)
      THEN Prtchar(byt)
      IF v+8<70 THEN v=v+8 FI FI
   FI
   under=Locate(v,w) color=hue(2)
   IF paint=2 THEN color=hue(3) FI
   Plot(v,w) Dly(1000) color=under
   Plot(v,w) Dly(1000) OD
RETURN

PROC Triangle()
 x1=x x2=x y1=y y2=y
DO color=0 Plot(x,y) DrawTo(x1,y1)
  DrawTo(x2,y2) DrawTo(x,y)
 stk= Stick(0) Hues() Dly(1000)
 IF (stk&1)=0 AND y1>1 THEN y1==-1 FI
```

```
    IF (stk&2)=0 AND y1<87 THEN y1==+1 FI
    IF (stk&8)=0 AND x2<78 AND Strig(0)=1
       THEN x2==+1 FI
    IF (stk&4)=0 AND x2>1 AND Strig(0)=1
       THEN x2==-1 FI
    IF Strig(0)=0 AND (stk&8)=0 AND x1<78
      THEN x1==+1 FI
    IF Strig(0)=0 AND (stk&4)=0 AND x1>1
      THEN x1==-1 FI
    color=hue(paint) Plot(x,y)
    DrawTo(x1,y1) DrawTo(x2,y2)
    DrawTo(x,y)
    IF key=33 THEN
      IF x1<>x OR x2<>x OR y1<>y OR y2<>y
      THEN under=color FI EXIT FI OD
  RETURN

  PROC Round()
    r=0
    DO color=0 Circle(r) Hues()
    stk=Stick(0)
    IF (stk&1)=0 AND r+x<78 AND x-r>1
       AND r+y<87 AND y-r>1 THEN
       r==+1 FI
    IF (stk&2)=0 AND r>0 THEN
       r==-1 FI
    color=hue(paint) Circle(r)
    IF key=33 THEN EXIT FI
    OD
  RETURN

  PROC CIO=$E456(BYTE areg,xreg)
  ; See page 137 of Action! manual

  PROC Loadfile()
    BYTE ARRAY filename
    BYTE iocb2cmd=$362
    CARD scrn,iocb2buf=$364,iocb2len=$368
    filename="D:JOHNX.PIC"
    key=255 byt=0
    DO IF key<255 THEN byt=Peek(65278+key)
         key=255 IF byt>47 AND byt<58 THEN
                   filename(7)=byt FI
      EXIT FI OD
    IF byt<48 OR byt>57 THEN RETURN FI
    OPEN(2,filename,4,0) iocb2cmd=7
    scrn=Peek(89)*256 scrn==+peek(88)
    iocb2buf=scrn
    iocb2len=352 CIO(0,$20) Close(2)
    under=Locate(x,y)
  RETURN

  PROC Savefile()
    BYTE ARRAY filename
    BYTE iocb2cmd=$362
    CARD scrn,iocb2buf=$364,iocb2len=$368
    filename="D:JOHNX.PIC"
    key=255 byt=0
  DO IF key<255 THEN byt=Peek(65278+key)
       key=255 IF byt>47 AND byt<58 THEN
                   filename(7)=byt FI
      EXIT FI OD
    IF byt<48 OR byt>57 THEN RETURN FI
    OPEN(2,filename,8,0) iocb2cmd=$0B
    scrn=Peek(89)*256 scrn==+peek(88)
    iocb2buf=scrn
    iocb2len=352 CIO(0,$20) Close(2)
  RETURN

  PROC Speed()
    Ding(30,600) key=255 byt=spd+48
    y=70 w=89 Prtchar(byt)
    DO UNTIL key<255 OD
    byt=Peek(65278+key)
    IF byt<49 OR byt>57 THEN Mode()
    RETURN FI
    spd=byt-48 Mode() RETURN

  PROC Main()
   Init() sel=0 DO
    IF sel<>0 THEN sel=0 Mode() FI
    IF key=33 THEN Brush()
    ELSEIF key=63 THEN sel=2 Mode()
           y=x w=y Letters()
    ELSEIF key=18 THEN sel=4 Mode()
           Round()
    ELSEIF key=40 THEN sel=1 Mode()
           Rectangle()
    ELSEIF key=45 THEN sel=3 Mode()
           Triangle()
    ELSEIF key=0 THEN sel=5 Mode()
           Loadfile()
    ELSEIF key=62 THEN sel=5 Mode()
           Savefile()
    ELSEIF key=39 THEN Speed()
    FI Key=255
    IF consol=3 THEN Init() FI
    Draw() Hues()
    OD
  RETURN
```

Fun with Art Picture Loader
by Harry Perkins, F.A.C.S.
Fresno, CA

```
10 REM ***FUN WITH ART PICTURE LOADER*
**
15 REM ***REV.12/83 FOR F.A.C.S. NEWSL
ETTER**
20 REM ***BY HARRY PERKINS***
25 SE.2,0,0:PRINT"%":GOS.29000
30 PRINT"ENTER PICTURE FILE TO LOAD":I
NPUT PIC$
35 GOS.29100
40 IF PEEK(764)<>28 THEN G.40
50 GOS.29400
60 SE.2,0,0
70 G.20
29000 REM --INT FOR A FUN WITH ART--
29010 MX=7:DIM CIO$(MX)
29015 FOR I=1 TO MX:READ J:CIO$(I)=CHR
$(J):NEXT I
29020 D.104,169,16,170,76,86,228
29022 MX=15:DIM PIC$(MX)
29025 DIM DLION$(MX)
29030 FOR I=1 TO MX:READ J:DLION$(I)=C
HR$(J):NEXT I
29035 D.104,169,192,141,232,6,162,6,16
0,221
29040 D.169,6,76,92,228
29045 MX=18:DIM DLIOFF$(MX)
29050 FOR I=1 TO MX:READ J:DLIOFF$(I)=
CHR$(J):NEXT I
29055 D.104,169,64,141,232,6,141,14,21
2,162
29060 D.228,160,95,169,6,76,92,228
29070 REM INIT. VARIABLES AND STEAL SO
ME MEMORY FROM BASIC
29075 REM
29080 IOCB=848:OLDSCL=PEEK(560):OLDSCH
=PEEK(561)
29085 PICBAS=(INT(PEEK(742)/16)-2)*16
29090 DLIBAS=PICBAS-9:POKE 741,0:POKE
742,DLIBAS
29095 RET.
29100 REM -- LOAD AND SHOW A FUN WITH
ART PICTURE--
29105 REM ROUTINE RETURNS A=-1 IF SOME
THING IS WRONG WITH FILE
```

RUTH'S PILOT (cont'd)

```
2030 LB .BYTE "SECTOR LINKS MANGLED",0
2040 DSKFULL .BYTE "DISK IS FULL",0
2050 CR .BYTE "CANT READ/WRITE SECTOR"
,0
2060 EOF .BYTE "END OF FILE ERROR ",0
2070 NUMERR .BYTE "BAD DRIVE #",0
2080 TO .BYTE "DISK TIMEOUT ERROR",0
2090 END = *+5
```

* * * LISTINGS FROM LAST MONTH

Greg Menke: Memory Drive

```
1 REM ********************************
2 REM **      ACE NEWSLETTER        **
3 REM **    3662 VINE MAPLE DR      **
4 REM **     EUGENE, OR 97405       **
5 REM **    MAY 1984  $12 YEAR      **
6 REM ********************************
7 REM **   BASIC LOADER PROGRAM     **
8 REM **    FOR MEMORY DRIVE        **
9 REM ** BY GREG MENKE-see text     **
      ** April ACE for details     **
      ********************************
10 GRAPHICS 0:SETCOLOR 2,12,4:? :? "Wr
iting D:MEMORY.LOD":? :? :?
20 OPEN #1,8,0,"D:MEMORY.LOD"
30 READ A:IF A<>-1 THEN PUT #1,A:GOTO
30
40 CLOSE #1:? "Complete...":? :END
999 END
1000 DATA 255,255,0,6,215,6,104,162,0,
189,26,3,201,0,240
1010 DATA 12,201,77,240,8,232,232,232,
224,38,144,238,96,169,77
1020 DATA 157,26,3,169,38,157,27,3,169
,6,157,28,3,96,85
1030 DATA 6,59,6,126,6,99,6,62,6,62,6,
76,0,6,0
1040 DATA 0,0,0,0,0,0,160,1,96,160,146
,96,160,136,96
1050 DATA 24,165,203,105,1,133,203,165
,204,105,0,133,204,96,160
1060 DATA 255,96,169,128,133,204,169,0
,133,203,141,59,6,76,60
1070 DATA 6,160,0,145,203,32,69,6,24,1
73,55,6,105,1,141
1080 DATA 55,6,173,56,6,105,0,141,56,6
,76,60,6,173,59
1090 DATA 6,201,0,208,188,173,55,6,201
,0,208,7,173,56,6
1100 DATA 201,0,240,191,160,0,177,203,
133,207,32,69,6,24,173
1110 DATA 57,6,105,1,141,57,6,173,58,6
,105,0,141,58,6
1120 DATA 173,55,6,205,57,6,208,27,173
,56,6,205,58,6,208
1130 DATA 19,169,0,141,55,6,141,56,6,1
41,57,6,141,58,6
1140 DATA 169,1,141,59,6,165,207,76,60
,6,0,0
1160 DATA -1
```

```
10 ;Atari Memory Drive by Greg Menke
20 ;
30 ;2/22/84 V1.0
40 ;
50 ;
60 ;Uses Atari Assembler/Editor
70 ;format.
80 ;
90 ;
0100 ;.WORD assembles label lo byte fi
rst into memory, then hi byte
0110 ;.BYTE assembles numbers followin
g into memory
0120 ;TABLE/256 gives hi byte of TABLE
0130 ;TABLE&255 gives lo byte of TABLE
0140 ;
0150  *=$600
0160  PLA
0170 START LDX #0
0180 FIND LDA $031A,X  ;get i byte fro
m HANDLER table
0190  CMP #0           ;is it free?
0200  BEQ INIT         ;yes, use this
one
0210  CMP #'M          ;have we used i
t before?
0220  BEQ INIT         ;yes, use this
one anyway
0230  INX              ;
0240  INX              ;go to next entry
0250  INX              ;
0260  CPX #$26         ;end of table?
0270  BCC FIND         ;no, keep going
0280  RTS              ;no more space
0290 INIT LDA #'M      ;our Device cod
e
0300  STA $031A,X      ;in HATABS
0310  LDA #TABLE&255   ;vector table l
o byte
0320  STA $031B,X
0330  LDA #TABLE/256   ;vector table h
i byte
0340  STA $031C,X
0350  RTS              ;return
0360 ;
0370 TABLE .WORD OPEN-1 ;OPEN vector
0380  .WORD NOERR-1     ;CLOSE vector
0390  .WORD READ-1      ;READ vector
0400  .WORD WRITE-1     ;WRITE vector
0410  .WORD NOFUNC-1    ;STATUS vector
0420  .WORD NOFUNC-1    ;SPECIAL vecto
r
0430  JMP START-1       ;Jump to initi
alizing
0440 ;
0450 VECTOR .BYTE 0,0  ;
0460 LEN .BYTE 0,0     ;some variabl
es
0470 LENSHADOW .BYTE 0,0 ;
0480 FLAG .BYTE 0      ;
0490 ;
0500 NOERR LDY #1      ;clear potenti
al error
0510  RTS              ;return
0520 NOFUNC LDY #146   ;function not
allowed error
0530  RTS              ;return
0540 DONE LDY #136     ;end of file
0550  RTS              ;return
0560 ADDONE CLC        ;
0570  LDA $CB          ;
0580  ADC #1           ;
0590  STA $CB          ;add one to
0600  LDA $CC          ;current file poi
nter
0610  ADC #0           ;
0620  STA $CC          ;
0630  RTS              ;
0640 ERROR LDY #255    ;home brewed e
rror
0650  RTS              ;return
0660 ;
0670 OPEN LDA #$80     ;
0680  STA $CC          ;clear addresses
0690  LDA #0           ;abort if error
0700  STA $CB          ;is detected
0710  STA FLAG         ;
0720  JMP NOERR        ;clear error
0730 ;
0740 WRITE LDY #0      ;
0750  STA ($CB),Y      ;
0760  JSR ADDONE       ;
0770  CLC              ;store byte of
0780  LDA LEN          ;file in memory
0790  ADC #1           ;and increment
0800  STA LEN          ;length
0810  LDA LEN+1        ;
0820  ADC #0           ;
0830  STA LEN+1        ;
0840  JMP NOERR        ;clear error
0850 ;
0860 READ LDA FLAG     ;
0870  CMP #0           ;
0880  BNE DONE         ;
0890  LDA LEN          ;
0900  CMP #0           ;
0910  BNE CONT         ;
0920  LDA LEN+1        ;
```

```
0930  CMP #0          ;
0940  BEQ ERROR
0950 CONT LDY #0      ;
0960  LDA ($CB),Y     ;
0970  STA $CF         ;get a byte of
0980  JSR ADDONE      ;file from
0990  CLC             ;memory, incremen
t
1000  LDA LEN5HADOW   ;address and chec
k for EOF
1010  ADC #1          ;
1020  STA LEN5HADOW    ;
1030  LDA LEN5HADOW+1  ;
1040  ADC #0          ;
1050  STA LEN5HADOW+1  ;
1060  LDA LEN          ;
1070  CMP LEN5HADOW    ;
1080  BNE OK           ;
1090  LDA LEN+1        ;
1100  CMP LEN5HADOW+1  ;
1110  BNE OK           ;
1120  LDA #0          ;
1130  STA LEN          ;
1140  STA LEN+1        ;
1150  STA LEN5HADOW    ;
1160  STA LEN5HADOW+1  ;
1170  LDA #1          ;
1180  STA FLAG         ;
1190 OK LDA $CF        ;restore Acc.
1200  JMP NOERR        ;clear error
```

More from Greg Menke:

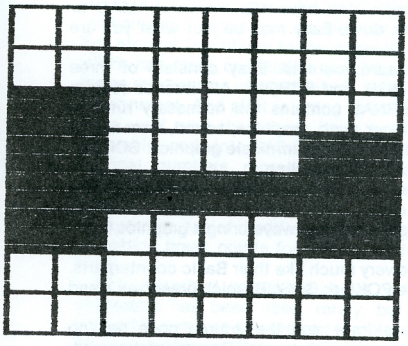
```
1 REM ********************************
2 REM **      ACE NEWSLETTER        **
3 REM **    3662 VINE MAPLE DR      **
4 REM **     EUGENE, OR 97405       **
5 REM **    MAY 1984  $12 YEAR      **
6 REM ********************************
7 REM **   NEWDOS BASIC LOADER      **
8 REM **      BY GREG MENKE         **
9 REM ********************************
10 GRAPHICS 0:SETCOLOR 2,0,0:? "Writin
g D:AUTORUN.SYS, please wait..."
20 OPEN #1,8,0,"D:AUTORUN.SYS"
30 READ A:IF A=-1 THEN 50
40 PUT #1,A:GOTO 30
50 CLOSE #1:? :? "Done":? :END
1000 DATA 255,255,252,28,123,31,169,65
,133,10,141,70,21,169,29
1010 DATA 133,11,141,74,21,169,32,141,
214,18,169,83,141,215,18
1020 DATA 169,29,141,216,18,169,31,141
,232,2,169,123,141,231,2
1030 DATA 32,49,29,96,0,32,252,28,32,4
9,29,108,250,191,169
1040 DATA 76,141,29,25,169,40,141,30,2
5,169,29,141,31,25,96
1050 DATA 169,157,141,214,18,169,67,14
1,215,18,169,3,141,216,18
1060 DATA 76,159,23,157,67,3,141,108,2
9,201,1,240,10,173,109
1070 DATA 29,201,0,208,3,76,139,29,173
,108,29,96,0,0,0
1080 DATA 0,162,31,160,45,76,24,30,162
,31,160,68,76,24,30
1090 DATA 162,31,160,87,76,24,30,162,3
1,160,99,76,24,30,173
1100 DATA 108,29,168,192,138,240,241,1
92,160,240,230,192,136,240,219
1110 DATA 192,144,240,206,192,173,240,
43,192,172,240,46,192,171,240
1120 DATA 49,192,170,240,52,192,169,24
0,55,192,168,240,58,192,167
1130 DATA 240,61,192,166,240,64,192,16
5,240,67,192,164,240,70,192
1140 DATA 162,240,73,76,103,29,162,30,
160,98,76,24,30,162,30
1150 DATA 160,111,76,24,30,162,30,160,
238,76,24,30,162,30,160
1160 DATA 162,76,24,30,162,30,160,137,
76,24,30,162,30,160,183
1170 DATA 76,24,30,162,30,160,252,76,2
4,30,162,30,160,203,76
1180 DATA 24,30,162,30,160,222,76,24,3
0,162,31,160,11,76,24
1190 DATA 30,162,31,160,32,134,204,132
,203,160,0,140,107,29,32
1200 DATA 67,30,172,107,29,238,107,29,
177,203,201,0,240,10,24
1210 DATA 109,110,29,32,78,30,76,36,30
,32,67,30,173,108,29
1220 DATA 76,103,29,169,155,32,78,30,1
69,155,32,78,30,96,172
1230 DATA 6,228,140,96,30,238,96,30,17
2,7,228,140,97,30,108
1240 DATA 96,30,0,0,70,79,82,77,65,84,
32,69,82,82,79
1250 DATA 82,0,65,80,80,69,78,68,32,68
,79,83,49,32,84
1260 DATA 79,32,68,79,83,50,32,69,82,8
2,79,82,0,68,73
1270 DATA 82,69,67,84,79,82,69,32,70,8
5,76,76,44,32,54
1280 DATA 52,32,70,73,76,69,83,0,70,73
,76,69,32,78,79
1290 DATA 84,32,70,79,85,78,68,32,69,8
2,82,79,82,0,73
1300 DATA 78,86,65,76,73,68,32,88,73,7
9,32,67,79,77,77
1310 DATA 65,78,68,0,80,79,73,78,84,32
,76,69,78,71,84
1320 DATA 72,32,69,82,82,79,82,0,70,73
,76,69,32,78,65
1330 DATA 77,69,32,69,82,82,79,82,0,80
,79,73,78,84,32
1340 DATA 73,78,86,65,76,73,68,0,70,73
,76,69,32,73,83
1350 DATA 32,76,79,67,75,69,68,0,83,69
,67,84,79,82,32
1360 DATA 76,73,78,75,83,32,77,65,78,7
1,76,69,68,0,68
1370 DATA 73,83,75,32,73,83,32,70,85,7
6,76,0,67,65,78
1380 DATA 84,32,82,69,65,68,47,87,82,7
3,84,69,32,83,69
1390 DATA 67,84,79,82,0,69,78,68,32,79
,70,32,70,73,76
1400 DATA 69,32,69,82,82,79,82,32,0,66
,65,68,32,68,82
1410 DATA 73,86,69,32,35,0,68,73,83,75
,32,84,73,77,69
1420 DATA 79,85,84,32,69,82,82,79,82,0
,0,0,0,0,0
1430 DATA 0,0,255,0,255,59,224,2,225,2
,252,28
1450 DATA -1
```

DOS Error Handler

```
10 ;DOS error handler by Greg Menke.
20 ;
30 ;V1.0 3/12/84
40 ;
50 ;
60 ;
70 ;
80 ;Uses Atari Assembler/Editor
90 ;format.
0100 ;
0110 ;
0120 ;
0130 ;
0140 ;POKE 7533 ($1D6D),0 - Print erro
r
0150 ;                   ,1 - Don't
0160 ;
0170 ;POKE 7534 ($1D6E),0 - Normal
0180 ;                   ,128 - Inverse
0190 ;
0200 ;
0210 ;
0220  *=$1CFC
0230 BEGIN LDA #NEWDOS&255
0240  STA 10
0250  STA 5446
0260  LDA #NEWDOS/256
0270  STA 11
0280  STA 5450
0290 ;
0300  LDA #$20        ;
0310  STA $12D6       ;hand compile a
```

```
0320  LDA #START&255 ;JSR to the
0330  STA $12D6+1    ;error handling
0340  LDA #START/256 ;routine into
0350  STA $12D6+2    ;DOS.
0360 ;
0370  LDA #END/256   ;
0380  STA 744        ;set LOMEM past
0390  LDA #END&255   ;end of
0400  STA 743        ;program.
0410  JSR INIT
0420  RTS
0430  BRK
0440 ;
0450 INIT2 JSR BEGIN
0460  JSR INIT
0470  JMP ($BFFA)
0480 INIT LDA #$4C   ;modify MEM.SAV
0490  STA $191D      ;routines to
0500  LDA #INIT2&255 ;JMP to program
0510  STA $191D+1    ;initialization.
0520  LDA #INIT2/256 ;
0530  STA $191D+2    ;
0540  RTS
0550 ;
0560 NEWDOS LDA #$9D ;reset the
0570  STA $12D6      ;modified DOS
0580  LDA #$43       ;back to normal
0590  STA $12D6+1    ;on DOS call.
0600  LDA #$3        ;
0610  STA $12D6+2    ;
0620 ;
0630  JMP 6047       ;load DOS.
0640 ;
0650 ;
0660 ;Error handling routine starts
0670 ;here.
0680 ;
0690 ;
0700 START STA $343,X ;code we repla
d in DOS.
0710  STA ERROR
0720  CMP #1
0730  BEQ EXIT
0740  LDA FLAG
0750  CMP #0
0760  BNE EXIT
0770  JMP CONT
0780 ;
0790 EXIT LDA ERROR
0800  RTS
0810 INDEX .BYTE 0
0820 ERROR .BYTE 0
0830 FLAG .BYTE 0
0840 INVERSE .BYTE 0
0850 ;
0860 CANTREAD LDX #CR/256
0870  LDY #CR&255
0880  JMP PRINT
0890 ENDOFILE LDX #EOF/256
0900  LDY #EOF&255
0910  JMP PRINT
0920 DRIVENUM LDX #NUMERR/256
0930  LDY #NUMERR&255
0940  JMP PRINT
0950 TIMEOUT LDX #TO/256
0960  LDY #TO&255
0970  JMP PRINT
0980 ;
0990 CONT LDA ERROR
1000  TAY
1010  CPY #138
1020  BEQ TIMEOUT
1030  CPY #160
1040  BEQ DRIVENUM
1050  CPY #136
1060  BEQ ENDOFILE
1070  CPY #144
1080  BEQ CANTREAD
1090  CPY #173
1100  BEQ BADSECTOR
1110  CPY #172
1120  BEQ APPEND1TO2
1130  CPY #171
1140  BEQ POINT
1150  CPY #170
1160  BEQ NOTFOUND
1170  CPY #169
1180  BEQ DIRFULL
1190  CPY #168
1200  BEQ CMDINVALID
1210  CPY #167
1220  BEQ FILELOCKED
1230  CPY #166
1240  BEQ LENGTH
1250  CPY #165
1260  BEQ FILERROR
1270  CPY #164
1280  BEQ LINK5
1290  CPY #162
1300  BEQ DISKFULL
1310  JMP EXIT
1320 ;
1330 BADSECTOR LDX #BS/256
1340  LDY #BS&255
1350  JMP PRINT
1360 APPEND1TO2 LDX #A1/256
1370  LDY #A1&255
1380  JMP PRINT
1390 POINT LDX #PI/256
1400  LDY #PI&255
1410  JMP PRINT
1420 NOTFOUND LDX #NF/256
1430  LDY #NF&255
1440  JMP PRINT
1450 DIRFULL LDX #DF/256
1460  LDY #DF&255
1470  JMP PRINT
1480 CMDINVALID LDX #CMP/256
1490  LDY #CMP&255
1500  JMP PRINT
1510 FILELOCKED LDX #LOCKED/256
1520  LDY #LOCKED&255
1530  JMP PRINT
1540 LENGTH LDX #DLE/256
1550  LDY #DLE&255
1560  JMP PRINT
1570 FILERROR LDX #FNE/256
1580  LDY #FNE&255
1590  JMP PRINT
1600 LINK5 LDX #LB/256
1610  LDY #LB&255
1620  JMP PRINT
1630 DISKFULL LDX #DSKFULL/256
1640  LDY #DSKFULL&255
1650 PRINT STX $CC
1660  STY $CB
1670  LDY #0
1680  STY INDEX
1690  JSR RETURN5
1700 LOOP LDY INDEX
1710  INC INDEX
1720  LDA ($CB),Y
1730  CMP #0
1740  BEQ DONE
1750  CLC
1760  ADC INVERSE
1770  JSR PRINTIT
1780  JMP LOOP
1790 DONE JSR RETURN5
1800  LDA ERROR
1810  JMP EXIT
1820 RETURN5 LDA #155
1830  JSR PRINTIT
1840  LDA #155
1850  JSR PRINTIT
1860  RTS
1870 PRINTIT LDY $E406 ;screen editor
1880  STY VECTOR       ;PUT byte
1890  INC VECTOR       ;routine +1
1900  LDY $E407        ;
1910  STY VECTOR+1     ;
1920  JMP (VECTOR)
1930 VECTOR .BYTE 0,0
1940 BS .BYTE "FORMAT ERROR",0
1950 A1 .BYTE "APPEND DOS1 TO DOS2 ERR
OR",0
1960 DF .BYTE "DIRECTORY FULL, 64 FILE
S",0
1970 NF .BYTE "FILE NOT FOUND ERROR",0
1980 CMP .BYTE "INVALID XIO COMMAND",0
1990 DLE .BYTE "POINT LENGTH ERROR",0
2000 FNE .BYTE "FILE NAME ERROR",0
2010 PI .BYTE "POINT INVALID",0
2020 LOCKED .BYTE "FILE IS LOCKED",0
```

Ruth Ellsworth: PILOT Character Maker



```
10 R:PILOT CHARACTER MAKER UTILITY
20 R:Ruth Ellsworth
30 R:ACE NEWSLETTER
40 R:MAY 1984  Eugene, Oregon
50 R:LINES 450 TO 470 FIND OUR LOCATIO
N IN THE ARRAY.  BECAUSE OUR GRID HAS
NINE NUMBERS BETWEEN 0 AND 8
60 R:USING 0 AS A STARTER, WE ARE ABLE
 TO LOCATE THE RELATIONSHIP OF EACH VA
LUE TO NUMBER ADDED
70 R:TO MEMORY LOCATION OFFSET.  IF TH
E CONSISTANT REALTIONSHIP OF LOCATIONS
 HAD NOT APPLIED
80 R:WE WOULD HAVE HAD TO USE THE LESS
MATHEMATICAL MEANS SHOWN IN THE R:EXAM
PLE MODULE AT THE END OF
90 R:THIS LISTING.
100 GR:CLEAR
110 C:#X=-70
120 C:#Y=35
130 C:##M=1540
140 U:*COUNTER
150 U:*GRID
160 J:*JOY
170 *COUNTER
180 C:@##M=0
190 C:##M=##M+1
200 J(##M(1604):*COUNTER
210 E:
220 *GRID
230 GR:GOTO-70,44
240 GR:4(TURN90;DRAW72)
250 GR:3(TURN90;G09;TURN90;DRAW72;TURN
-90;G09;TURN-90;DRAW72)
260 GR:TURN90;G09;TURN90;DRAW72
270 GR:TURN-90;G09;TURN-90;G09;TURN-90
;DRAW72
280 GR:3(TURN90;G09;TURN90;DRAW72;TURN
-90;G09;TURN-90;DRAW72)
290 GR:TURNT00
300 E:
310 *JOY
320 C(%J0=1):#Y=#Y+9
330 C(%J0=8):#X=#X+9
340 C(%J0=2):#Y=#Y-9
350 C(%J0=4):#X=#X-9
360 C(#X(-70):#X=-70
370 C(#X)-7):#X=-7
380 C(#Y(-28):#Y=-28
390 C(#Y)35):#Y=35
400 GR:GOTO#X,#Y
410 GR:PENRED;4(DRAW9;TURN90)
420 GR:PENYELLOW;4(DRAW9;TURN90)
430 J(@B764=18):*CHAR
440 J(@B764=42):*ERASE
450 J(%T8=1):*FILL
460 J(%T8=0):*JOY
470 E:
480 *FILL
490 GR:FILL9
500 C:#C=(#X-2)/9+8[Column location
510 C:#R=(#Y+1)/9+3[Row location
520 C:#O=#R*8+#C[Memory offset
530 C:#V=1540+#O[1540 has to be used i
n this equation not #M
540 C(#C=0):@##V=128
550 C(#C=1):@##V=64
560 C(#C=2):@##V=32
570 C(#C=3):@##V=16
580 C(#C=4):@##V=8
590 C(#C=5):@##V=4
600 C(#C=6):@##V=2
610 C(#C=7):@##V=1
620 J:*JOY
630 E:
640 *ERASE
650 GR:FILL9
660 C:#C=(#X-2)/9+8
670 C:#R=(#Y+1)/9+3
680 C:#O=#R*8+#C
690 C:#V=1540+#O
700 C:@##V=0
710 GR:G09;TURN90;G01;TURN90
720 GR:PENERASE;DRAW8;TURN-90;DRAW1;TU
RN-90;DRAW7;3(TURN90;G01;TURN90;DRAW7;
TURN-90;G01;TURN-90;DRAW7)
730 GR:DRAW-7;PENYELLOW;GO-1;TURN-90;D
RAW8;TURN90
740 C:@B764=49
750 J:*JOY
760 E:
770 *CHAR
780 C:##=@B1540+@B1541+@B1542+@B1543+@
B1544+@B1545+@B1546+@B1547
790 C:#G=@B1548+@B1549+@B1550+@B1551+@
B1552+@B1553+@B1554+@B1555
800 C:#F=@B1556+@B1557+@B1558+@B1559+@
B1560+@B1561+@B1562+@B1563
810 C:#E=@B1564+@B1565+@B1566+@B1567+@
B1568+@B1569+@B1570+@B1571
820 C:#D=@B1572+@B1573+@B1574+@B1575+@
B1576+@B1577+@B1578+@B1579
830 C:#C=@B1580+@B1581+@B1582+@B1583+@
B1584+@B1585+@B1586+@B1587
840 C:#B=@B1588+@B1589+@B1590+@B1591+@
B1592+@B1593+@B1594+@B1595
850 C:#A=@B1596+@B1597+@B1598+@B1599+@
B1600+@B1601+@B1602+@B1603
860 T:ROW 1 = #A   ROW 2 = #B   ROW 3
= #C
870 T:ROW 4 = #D   ROW 5 = #E   ROW 6
= #F
880 T:ROW 7 = #G   ROW 8 = #H
890 WRITE:D:MAKER,#A
900 WRITE:D:MAKER,#B
910 WRITE:D:MAKER,#C
920 WRITE:D:MAKER,#D
930 WRITE:D:MAKER,#E
940 WRITE:D:MAKER,#F
950 WRITE:D:MAKER,#G
960 WRITE:D:MAKER,#H
970 CLOSE:D:MAKER
980 PA:400
990 E:
1000 R:EXAMPLE
1010 R:J(#Y=35):*ROW1(LOCATES ROW
1020 R:THESE LINES TO LOCATE ROWS 1 TO
 8 WOULD BE FOUND IN THE FILL MODULE
1030 R:C(#X=-70):@B1540=128(LOCATES CO
LUMN
1040 R:THESE LINES WOULD BE FOUND IN *
ROW MODULES
```

```
10 R:CHARACTER RETRIEVAL ROUTINE
20 READ:D:MAKER,#A
30 READ:D:MAKER,#B
40 READ:D:MAKER,#C
50 READ:D:MAKER,#D
60 READ:D:MAKER,#E
70 READ:D:MAKER,#F
80 READ:D:MAKER,#G
90 READ:D:MAKER,#H
100 CLOSE:D:MAKER
110 E:
```

```
5 R:ANTIC AUGUST 1983
7 R:PHIL AND KATHY BERGH
10 GR:QUIT
20 T:Please wait while the character s
et is moved.  This takes about 20 seco
nds.
```

GOTO PAGE 9

JOHNNY'S PAINTBOX

Johnny's Paintbox is a drawing program offering six colors and a number of automatic shape generators. In addition to the normal line drawing mode there are modes to draw Rectangles, Triangles, Circles the Alphabet as well as modes to Save and Load pictures.

NORMAL MODE

All other modes return to the normal line drawing mode. Lines are drawn using a joystick in player #1 position. The trigger must be pressed to cause the line to be drawn. This mode is indicated by a horizontal line to the right of the numbers at the bottom of the screen. The color of the line indicates the color active at the moment. You can select a new color by pressing the number key corresponding to the desired color. This is also true for the shape modes. Color 0 can be used to erase lines, (again, the trigger must be pressed).

The 'OPTION' key can be used to clear the screen. It is only active in the normal mode. There is also a 'fill' function in the normal mode. It is activated by pressing the spacebar. An area enclosing the cursor will be filled in using the color currently active. Sometimes only part of the area will be filled in and it will be necessary to move to a blank portion and use the spacebar again. Only background area can be filled in.

The speed of the cursor can be changed using the key with the Atari symbol. When you press it, a number representing the current speed will appear. You may now select any number 1-9 with 1 being the fastest. Any other key will return you to normal mode with the speed unchanged.

CIRCLE MODE

The circle mode is entered by pressing the 'C' key. A circle in the active color will appear to the right of the numbers. Position the cursor where the circle center is to be located before you press 'C'. Pushing the joystick up will increase the size and down will decrease the size of the circle. If you return to the original cursor size, no circle will be drawn. If any part of the circle reaches an edge you will not be allowed to increase its size further. Change the circle color by using the number keys. The circles are actually ovals because of the difference in horizontal and vertical sizes of the pixels. Press the spacebar to exit the circle mode when you have the size of circle you desire.

RECTANGLES

Use the 'R' key to enter this mode. The cursor must be positioned where the lower left corner of the rectangle will be. Vertical stick movement incereases and decreases the vertical size of the rectangle while horizontal movement controls the horizontal size. Number keys allow you to change the color and the spacebar is used to freeze the rectangle and exit the mode. This mode is useful to erase blocks of area on the screen. Just expand the rectangle out and contract it back to a point before hitting the spacebar.

TRIANGLES

Enter using the 'T' key. Horizontal joystick movement, (without the button pushed), controls the baseline of the triangle while vertical movement controls the vertical position of the vertex. The starting position is at the leftmost edge of the baseline. By pushing the fire button while moving the stick horizontally, you can control the horizontal position of the vertex. This mode is a little trickier than the others and may take some practice. Again, use the spacebar to exit.

ALPHABET

Uppercase letters of the alphabet can be drawn in this mode. The starting position indicates the upper left corner of the first letter. Enter using the 'A' key. An 'A' in the active color indicates you are in the alphabet mode. The cursor will move after each letter is drawn. If you want to go back, the backspace key works. Again, number keys can be used to change the current color. For this reason, you can't draw numbers, just letters. The program won't let you draw letters if your position is too far to the right or too low for complete letters to be drawn. You must use the RETURN key to leave this mode since the space is a valid character.

SAVE MODE

When you have finished your sketch and wish to save it to disk, press the 'S' key. A question mark will appear to the right of the numbers. It is asking for a filenumber. You have a choice of ten from 0 thru 9. Press one of these number keys to save your picture. If a picture of that number already exists it will be written over. You will automatically be returned to the normal mode when the save is complete. Pressing any key other than 0-9 will abort the save.

LOAD MODE

Press 'L' to load a file. The question mark indicates you should press a number 0-9 as the filenumber. The program will load the file to the screen and return to the normal mode. There is no provision to allow for a file not existing, an error condition will occur and stop the program. To prevent this you should save something under each filenumber, even if it is a blank screen.

— Stan Ockers

EASY

MAC/65 and AMAC (Atari's Macro Assembler) owners, **EASY** ($40 from Superware, 2028 Kingshouse Rd., Silver Spring, MD 20904) is here to help you. If you know Basic well and have just a smattering of Assembly Language experience, and have 48k memory, AMAC or MAC/65 Assemblers, and disk drive, Easy may be just what you are looking for.

Easy is a collection of Macro routines. Easy consists of three libraries labeled KERNAL, PMGR, and SCROLL. All libraries require KERNAL to be included. KERNAL contains the necessary runtime code, it also contains I/O, integer math, graphics, sound, error handling, etc. PMGR contains routines for player/missle graphics. SCROLL contains routines for automatic fine scrolling.

The runtime code becomes part of your routine which can be a blessing for debugging your code. On the other hand much of the runtime code is not really necessary. Errors always bring a graphics 0 and an error number.

Many of the commands are very much like their Basic counterparts. Macros range from the simple POKE to GOSUB which preserves X and Y registers.

Documentation is a little skimpy and the source code has no remarks to let you know what's going on or why. Documentation consists of a 5½ X 8-inch booklet and a quick reference card. The booklet gives a brief explanation of requirements and lists some simple examples for each macro showing the expanded code. I found the instruction booklet a bit awkward to use while typing as it is stapled rather than bound or looseleaf. I finally resorted to a paperpunch and small notebook I had available. It improved the booklet's usefulness immensely. The macros are scattered through the 57 pages without much order. I like to see more remarks in the source code.

I had some problems with a couple of the macros. One in particular was the drawto macro. After plotting a point the drawto line began one position to the right of the plotted point. When I looked in the code I found the macro moved the current X,Y position which was unnecessary and gave the wrong starting position.

The scrolling macros are relativly easy to implement and work very well.

I believe this package will be a good tool for the beginner but as a serious development tool it has some limitations.

— Chuck Ross

PHAROAH'S TOMB

You have been searching for years and have finally found the tomb of Ramakanaman. But you have to find a series of keys to enable you to enter the treasure chamber. Once you have reached the treasure chamber, you are given a chance to enter another and then another, with each one requiring more keys to unlock.

Use joystick 1 to control the man appearing at the top left of the screen. Move him to pick up the key appearing at a random position in the maze. Once you have found the key you must take it and deposit it at the door of the treasure chamber and go and get another until you have found all the keys required for the door.

You will see a line of keys under the game's name at the top of the screen. This shows you how many you must find in the current level.

Five enemies materialize from certain points in the maze. They wander around trying to protect the treasure. These enemies are birds, serpents and demons. Every so often one of these goes berserk and dashes around the maze. If the berserker touches another enemy, the touched one will disappear temporarily and may help you get by. But be careful, when the mad creature calms down the invisible ones will reappear.

You have two forms of protection: 1. Teleport: this will reposition you at a random spot in the maze; and, 2. Super Splatter: this will make all the creatures disappear for a moment. Each time you resort to protection adds another key you must get to unlock the door. If you ever get 18 keys across the top of the screen, the game ends. A T or S at screen top right indicates the protection mode. You may toggle between them with the SELECT key. To use your protection, push the Space Bar.

Pressing the fire button will freeze the action so you may scroll around the screen to plan your path, find the key, or find your man after Teleporting.

— Sydney Brown

SUPERLOAD

(reprinted from the April '84 issue of the Atari Computer Club of OKC)

Presented here is a method for loading assembler programs from within a BASIC program. Instantly. That's right — no more irritating "Initializing..." messages on the screen while data is read and poked into storage. In addition to this capability, you will also have a USR call which can move or initialize a multi-page area of storage — the two major reasons for using machine language programming in a BASIC environment. And all it is going to cost you is 100 bytes of storage.

Up until now there have been two popular methods for loading assembler code into BASIC programs: reading DATA statements with decimal numbers representing object code, and reading DATA statements with hexadecimal numbers representing object code. The second method is more efficient as far as storage is concerned, and also permits easier modification (as long as it's extremely minor, such as setting break points for testing). Unfortunately, both methods are slow.

A third method, letting the object code be represented by ATASCII characters has been used, rarely, but is actually a far preferable method in terms of storage and load speed. However, this method has its own problems. Several ATASCII characters, such as the quote character and the EOL character, cannot appear in the string data. There is a variation of this method: retaining the object code on disk and reading it into the string at run time; however, this is extremely cumbersome for cassette users.

Fortunately, a fourth scheme has recently arrived: the code can be "hidden" between two BASIC tables. I refer you to "Atari Safe RAM" in the October, 1983 issue of **COMPUTE**. I have chosen this method to introduce a utility I've found useful. I'll give you a 100 byte machine language program which is extremely easy to install. You can then call this routine to load your own assembler programs using the hexadecimal digit method — only without any perceptible delay.

Listing One is the program which will create another BASIC program on either disk or cassette which inserts SUPERLOAD into a current program in memory. If you are a cassette user, change Line 80 to: OPEN #1,8,0,"C:". When you enter and run this program, you're ready for an example of how to utilize SUPERLOAD.

Enter the program in Listing Two, but be sure to save it before proceeding. Note also in Listing Two at Line 10 that you will have to dimension a string, ASM$, which will be used as a buffer string. Now, all you have to do to insert SUPERLOAD into the program is type: E."D:SLDATA"(disk users), or E."C:"(cassette users). Several BASIC statements will be read in and executed in immediate mode, rearranging several BASIC pointers to protect SUPERLOAD, which is inserted at the same time. This is a permanent insertion — if you now save the program normally, SUPERLOAD will be retained whenever you reload the program. In other words, you only have to perform the ENTER procedure once. From the point at which the program is saved, no further action is required.

Upon execution, a message appears in the upper left hand corner of the TV screen, repeating itself diagonally towards the bottom. This displays two of the functions of SUPERLOAD. Line 40 contains the message displayed on the screen in hexadecimal format. Line 50 packs the data at the location into variable ADSCR — the address of screen memory. If nothing happens, or the word "superload" does not appear, you must go back to Listing One and check each line again.

The USR call has the following format: X = USR(ADML,ADSTR,ADDEST,STRLL,OPTION). Where ADML is the address of ML$; ADSTR is the address of the input data; ADDEST is the address of the output location; STRLL is the length of the input data. If OPTION is 0 STRLL cannot exceed 255. This is the hexadecimal pack, as in Line 40. If OPTION is 1 STRLL cannot exceed 65535. No packing occurs and the string is moved as is, as in Line 60.

If the above procedure checks out, then enter the following in immediate mode: POKE ADSCR,119:X = USR(ADML,ADSCR,ADSCR + 1,959,1). The entire screen should fill with the character "w". This initialization routine is actually a move routine trick which has been used by assembler programmers for years. The first POKE acts as a "seed" and the move simply propagates this byte into the next and succeeding locations.

Here are a couple of more tricks you can do with SUPERLOAD. Instead of the following statement: ADSCR = PEEK(88) + 256*PEEK(89), use ADSCR = USR(ADML,88,212,2,1). Here's how it works. The two bytes beginning at location 88 are moved to locations 212 and 213. These locations are reserved by BASIC for a return value to be passed back to a BASIC variable after a USR call. So, as long as you move two bytes to 212 and 213, you can assign any low byte-high byte pair to any BASIC variable!

Here's another one — instead of using the following statement to divide a variable into low and high bytes: HI = INT(X/256):LO = X-HI*256, use: Y = USR(ADML,X,0,0,1). This one is a little more subtle. Notice the length parameter is 0. All SUPERLOAD does is pull the parameters off the stack, storing them into their assigned storage locations, and executes a return. But since X was passed as a parameter, even though it is not necessarily an address, it will be divided into low and high byte form and stored at the parm1 memory locations — 96 and 97. So, after the last example is executed, the low byte of X can be retrieved by PEEK(96) and the high byte by PEEK(97). Y is a dummy variable only. Both of these last examples are more efficient than BASIC and execute faster. I offer them as examples only — it is up to you to decide if they are easier to use than their BASIC counterparts.

The most efficient utilization of the program load feature is seen when you use a BASIC string to contain the object code. If you use the same value for ADSTR and ADDEST — namely, the address of the string — the resulting object code will be packed into the first half of the string and will then be directly executable. But the program listing is unchanged. You therefore have a string which you can list on the screen and alter with the editory, but when run is object code!

If you'll type in Listing Three you can modify Listing Two to see what I mean. This program copies the character set (at Line 70) and modifies the character passed via a USR to be a different color. The method used is artifacting — every other pixel on the screen is bypassed. Note the string defined by Lines 40 and 45 is executed directly by line 75 — after the hexadecimal code has been packed into object code by Line 50.

Well, there it is. I'm sure you'll find many more uses for SUPERLOAD besides the ones I've illustrated. And remember all you future software authors — no more "Initializing..." messages!

— Terry Barker

T.A.C.

T.A.C. (Tactical Armor Command, Avalon Hill, $30.00) is a World War II armored simulation. You choose tanks, field guns, and infantry from any one of four countries (U.S., Great Britain, Germany, and Russia). You direct their attack upon the enemy forces.

From the menu you choose one or two players, skill level (if playing the computer), any one of five scenarios, and the points each player gets for his forces (from 12 to 224).

The game is a stepped-up version of the TANKTICS game (also by AVALON HILL). This time, the computer not only does all the computations, but controls the map as well. Each turn consists of several parts. The first part is tactical sighting. This is when the "active" unit looks for all units in its line of sight (friendly and enemy). Next is the strategic sighting. Now the tactical sighting is put on the overall map. Then comes the movement phase. Players set the speed for their units. Then comes the indirect fire phase. Armor units may fire indirectly at enemy units (both seen and unseen). Infantry and field guns may load/unload in this phase. Then comes the second part of the movement phase. When a player presses the button, the unit begins to move. Players can turn their units by moving the joystick, but it is very easy to oversteer.

I love the indirect fire mode. With a squadron of tanks, a skilled player can wipe out most slow tanks, all dismounted field guns, and infantry halftracks. In the engagement scenario, a player can lose half his force in a single turn. But to keep a player from losing interest too soon, there are four skill levels and four other scenarios. These range from static defense to stalemate. In some scenarios there are minefields to worry about, besides the other player.

The graphics are excellent. The scrolling is outstanding. There is some sound, but this is not the same quality as the graphics. The game designer must have spent some time collecting all the information on almost 40 different tanks. The computer determines the effectiveness of a shot by computing the armor thickness, weapon caliber, time tracked, and speed.

Overall, I think this is very good. But as a Squad Leader fan, I find a maximum of eight units constricting. And the map is not very large once a main battle tank gets rolling. This is one of the best wargames I've seen in a long time.

—Aaron Ness

Atari Computer Enthusiasts

A.C.E. is an independent, non-profit and tax exempt computer club and user's group with no connection to the Atari Company, a division of Warner Communication Company. We are a group interested in educating our members in the use of the Atari Computer and in giving the latest News, Reviews and Rumors.

All our articles, reviews and programs come from you, our members.

Our membership is world-wide; membership fees include the A.C.E. Newsletter. Dues are $12 a year for U.S., and $22 a year Overseas Airmail and include about 10 issues a year of the ACE Newsletter.
Subscription Dep't: 3662 Vine Maple Dr., Eugene, OR 97405.

President— Kirt Stockwell, 4325 Sean , Eugene, OR 97402 503-689-5355
Vice Pres— Larry Gold, 1927 McLean Blvd., Eugene, Or 97405 503-686-1490
Secretary— Bruce Ebling, 1501 River Loop #1, Eugene, OR 97404 503-688-6872
Librarian— Ron and Aaron Ness, 374 Blackfoot, Eugene, Or 97404 (503)689-7106.
Editors— Mike Dunn, 3662 Vine Maple Dr., Eugene, Or 97405 503-344-6193
Jim Bumpas, 4405 Dillard Rd., Eugene, Or 97405 503-484-9925
E.R.A.C.E. (Education SIG Editor)-Ali Erickson, 295 Foxtail Dr., Eugene, OR 97405 /503-687-1133
E.R.A.C.E. Corresponding Secretary-Robert Browning, 90 W. Myoak, Eugene, OR 97404 (503)689-1513.

Send 50c stamps or coin ($1 overseas) to the Ness' for the new, updated ACE Library List —new in May 84 !

Best of ACE books

Volume 1 contains bound issues of the ACE Newsletter from the first issue, Oct 81 to June of 1982

Volume 2 covers July 1982 to June 1983

Only $12 each ($2 extra for Airmail). Available only from:

George Suetsugu
45-602 Apuapu St
Kanoehe, HI 96744

SortFinder 1.2
A composite index of Atari related articles from 5 popular computer periodicals from Apr '81 to June '83, including ACE. Only $6 for ACE member from:
Jim Carr, Valley Soft
2660 S.W. DeArmond
Corvallis, OR 97333

ATARI COMPUTER ENTHUSIASTS
3662 Vine Maple Dr. Eugene OR 97405

FIRST CLASS MAIL

Bulletin Board
(503) 343-4352
On line 24 hours a day, except for servicing and updating. Consists of a Tara equipped 48K Atari 400 with a TARA keyboard, 2 double-density double sided disk drives with an ATR 8000 interface, 2- 8" double density disk drives, an Epson MX80 printer, a Hayes SmartModem; running the ARMUDIC Bulletin Board software written by Frank L. Huband, 1206 N. Stafford St., Arlington, VA 22201. See the Nov '82 issue for complete details.